T.C.
MİLLİ EĞİTİM BAKANLIĞI

MEGEP

(MESLEKÎ EĞİTİM VE ÖĞRETİM SİSTEMİNİN
GÜÇLENDİRİLMESİ PROJESİ)

BİLİŞİM TEKNOLOJİLERİ

NESNE TABANLI PROGRAMLAMA 3

ANKARA 2007

Milli Eğitim Bakanlığı tarafından geliştirilen modüller;

- Talim ve Terbiye Kurulu Başkanlığının 02.06.2006 tarih ve 269 sayılı Kararı ile onaylanan, Mesleki ve Teknik Eğitim Okul ve Kurumlarında kademeli olarak yaygınlaştırılan 42 alan ve 192 dala ait çerçeve öğretim programlarında amaçlanan mesleki yeterlikleri kazandırmaya yönelik geliştirilmiş öğretim materyalleridir (Ders Notlarıdır).
- Modüller, bireylere mesleki yeterlik kazandırmak ve bireysel öğrenmeye rehberlik etmek amacıyla öğrenme materyali olarak hazırlanmış, denenmek ve geliştirilmek üzere Mesleki ve Teknik Eğitim Okul ve Kurumlarında uygulanmaya başlanmıştır.
- Modüller teknolojik gelişmelere paralel olarak, amaçlanan yeterliği kazandırmak koşulu ile eğitim öğretim sırasında geliştirilebilir ve yapılması önerilen değişiklikler Bakanlıkta ilgili birime bildirilir.
- Örgün ve yaygın eğitim kurumları, işletmeler ve kendi kendine mesleki yeterlik kazanmak isteyen bireyler modüllere internet üzerinden ulaşılabilirler.
- Basılmış modüller, eğitim kurumlarında öğrencilere ücretsiz olarak dağıtılır.
- Modüller hiçbir şekilde ticari amaçla kullanılamaz ve ücret karşılığında satılamaz.

İÇİNDEKİLER

AÇIKLAMALAR

KOD	**482BK0076**
ALAN	**Bilişim Teknolojileri**
DAL/MESLEK	**Veritabanı Programcılığı**
MODÜLÜN ADI	**Nesne Tabanlı Programlama 3**
MODÜLÜN TANIMI	Sabitler ve değişkenlerin program içinde kullanımı, dizi kullanarak program yazma, temsilci ve olayları programda kullanma, nesne tabanlı programlamanın temel prensiplerini kod yazımında kullanmayla ilgili öğrenme materyalidir.
SÜRE	40/32
ÖN KOŞUL	Nesne Tabanlı Programlama 2 modülünü bitirmiş olmak
YETERLİK	Nesne tabanlı programlama dilinde kod yazmak
MODÜLÜN AMACI	**Genel Amaç** Gerekli ortam sağlandığında, değişkeni ve diziyi tanımlayabilecek, miras işlemlerini yapabileceksiniz. **Amaçlar** 1. Sabitler ve değişkenleri kullanabileceksiniz. 2. Dizi mantığını ve dizi değişkenlerini tanımlayabileceksiniz. 3. Üzerine bindirme üyesi oluşturabilecek ve miras alma işlemlerini yapabileceksiniz. 4. Temsilci ve olayları tanımlayabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	**Ortam** Atölye, laboratuvar, ev, bilgi teknolojileri ortamı (İnternet) vb, kendi kendinize veya grupla çalışabileceğiniz tüm ortamlar. **Donanım** Programlama dilini çalıştırabilecek yeterlikte bilgisayar, yedekleme için gerekli donanım (CD yazıcı, flash bellek), raporlama için yazıcı, sayfa için internet bağlantısı, kâğıt ve kalem.
ÖLÇME VE DEĞERLENDİRME	Modülün içinde yer alan her öğrenme faaliyetinden sonra verilen ölçme araçları ile kendinizi değerlendireceksiniz. Modül sonunda ise, bilgi ve beceriyi belirlemek amacıyla, öğretmeniniz tarafından belirlenecek ölçme aracıyla değerlendirileceksiniz.

GİRİŞ

Sevgili Öğrenci,

Okul yaşantınızda öğreneceğiniz her konu, yaptığınız uygulama ve tamamladığınız her modül bilgi dağarcığınızı geliştirecek ve ilerde atılacağınız iş yaşantınızda size başarı olarak geri dönecektir. Eğitim sürecinde daha öz verili çalışır ve çalışma disiplinini kazanırsanız; başarılı olmamanız için hiçbir neden yoktur.

Günümüzde Windows tabanlı görsel programlama dillerinin hızla gelişmekte olduğu ve kullanımının oldukça yaygınlaştığı görülmektedir. Bu programlama dilleri ile sizler programlama mantığını ve becerisini çok daha kolay kavrayacaksınız.

Bu modülle, tüm programlama dillerinde kullanılan değişken ve dizi kavramını öğreneceksiniz. Ayrıca, temsilci ve olayları tanımlayabilmeyi, bir metotla farklı ortamlarda ve nesnelerde değişik sonuçlar üretebilmeyi, bir sınıfın metotlarını kullanarak başka sınıflar türetebilmeyi de öğreneceksiniz.

Bu modülde anlatılan konuların tümünü öğrendiğinizde, nesne tabanlı programlama dilinin temelini öğrenmiş olacak ve kendinize göre basit programlar yapabileceksiniz.

ÖĞRENME FAALİYETİ-1

AMAÇ

Sabitler ve değişkenleri program içerisinde uygun şekilde tanımlayıp kullanabileceksiniz.

ARAŞTIRMA

- Değişkenlere neden ihtiyaç duyulur? Araştırınız.
- Farklı türlerdeki değişkenler neden kullanılır? Araştırınız.

1. SABİTLER VE DEĞİŞKENLER

Değişken, verilerin bellekte geçici olarak kaydedilmesini ve gerektiğinde kullanılmasını sağlayan değerdir. Nesne tabanlı programlama dilinde değişken kullanımı diğer programlama dillerindeki değişken kullanımlarıyla aynıdır.

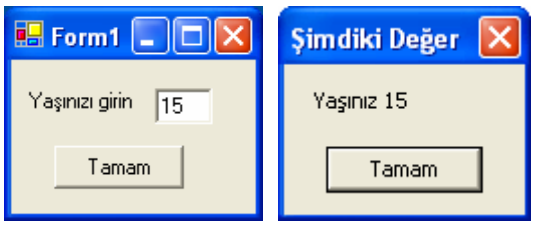
Bir değişkenin değeri program her çalıştırıldığında değişir. Örneğin, kişiye yaşını soruyorsak bu bilgiyi gerektiğinde program içinde kullanabilmek için bir değişkene aktarmalıyız. Değişkenler bellekte (RAM) yer kaplar. Yaş bilgisine 18 değeri girildiğinde bellekte yaş bilgisi için ayrılan alanda bu değer saklanır.

Örnek

Programın her çalışmasında kullanılan değişkenin aldığı değerin değiştiğini görmek istersek aşağıdaki program parçasını inceleyelim.

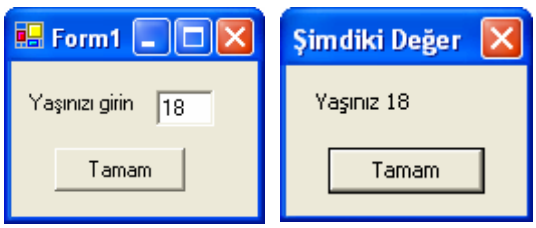
```
private void button1_Click(object sender, System.EventArgs e)
{
    string yaş;
    yaş=textBox1.Text ;
    MessageBox.Show ("Yaşınız "+yaş,"Şimdiki Değer");
}
```

Bu işlem için form tasarımında formun üzerine bir metin kutusu ve buton eklemeniz gerekir. Program parçası ilk çalıştırıldığında ekran görüntüsü resim 1.1'deki gibi olur.



Resim 1.1: Programın ilk çalışma görüntüsü

Programın çalışması sonlandırılıp yeniden çalıştırıldığında, daha önce girdiğimiz 15 değeri bellekten silinecek ve yeniden bir değer girmemiz gerekecektir (Resim1.2).



Resim 1.2: Yeni değerin görüntülenmesi

Bu işlem sonrasında değişkenimizin yeni değeri 18 olacaktır.

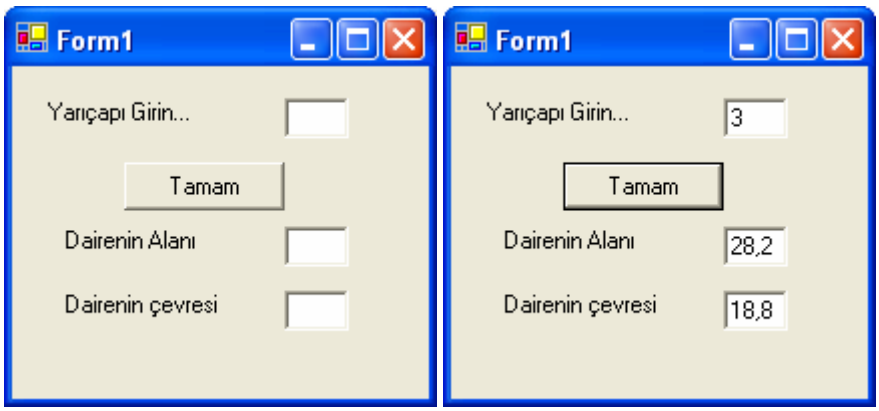
Sabitler, program içinde değeri değişmeyen ifadelerdir. Programda sabit tanımlandığında ilk değer mutlaka verilmelidir. Aksi taktirde programın çalışmasında hata oluşur. Sabitler programda **Const** deyimiyle tanımlanır.

Örneğin, pi sayısının programın tümünde aynı değerde olmasını istiyorsunuz. Pi sayısını Const tanımlama bloğunda 3.141592654 olarak belirttiğinizde bu değer programın her yerinde geçerli olacaktır.

Örnek

Butona (button) tıklandığında bir dairenin alanını ve çevresini metin kutusundan (textbox) girilen yarıçapa göre hesaplatan program kodu ve ekran çıktısı resim 1.3'teki gibidir.

```
private void button1_Click(object sender, System.EventArgs e)
{
    const float pi=3.141592654f;
    float alan,cevre;
    int r;
    r=Convert.ToInt32 (textBox1.Text);
    alan=pi*r*r;
    cevre=2*pi*r;
    textBox2.Text=alan.ToString ();
    textBox3.Text=cevre.ToString();
}
```



Resim 1.3: Daire alanının hesaplanması örneği ekran görüntüsü

1.1. Değişken Türleri

Değişkenler, sakladıkları bilgilerin türüne göre değişik şekillerde tanımlanır. Değişkenler tanımlanırken değişken türü, değişken adından önce yazılmalıdır. Nesne tabanlı programlamada kullanılan değişken türleri ve değer aralıkları tablo 1.1'de verilmiştir. Sadece bu programlama diline ait değişkenler vardır. Bunlar, sbyte, ulong, uint ve ushort değişken türleridir.

Değişken türleri	**Boyut (Byte)**	**Değer Aralığı**
byte	1	0-255
short	2	-32768 - 32767
int	4	-2.147.483.648 - 2.147.483.647
long	8	- 9.223.372.036.854.775.808 - 9.223.372.036.854.775.807
sbyte	1	-128 - 127
ushort	2	0 - 65535
uint	4	0 - 4.294.967.295
ulong	8	0 - 18.446.744.073.709.551.615
float	4	$\pm 1.5x10^{-45}$ - $\pm 3.4x10^{38}$
double	8	$\pm 5.0x10^{-324}$ - $\pm 1.7x10^{308}$
object		
char	2	Bir Unicode karakter
string		Karakterlerin tümü
decimal	16	$\pm 1.0 \times 10^{-28}$ to $\pm 7.9 \times 10^{28}$
bool	2	True veya False

Tablo 1.1: Değişken türlerinin boyut ve kapasiteleri

1.1.1. Tamsayı Değişken Türleri

1.1.1.1. Byte ve Sbyte Değişken Türleri

Byte, 0-255 arasında değer alabilen bir değişken türüdür. 1 byte boyutunda bilgiyi saklar. Sbyte ise, -128 +127 arasında değer alabilir. Yine 1 byte boyutunda bilgi saklar. Sbyte değişken türünün byte değişken türünden farkı, negatif sayıları saklıyor olmasıdır. Sbyte'taki S harfi signed (işaretli) anlamındadır.

```
private void Form1_Load(object sender, System.EventArgs e)
{
    sbyte sayi;
    sayi=50;
    sayi=(sbyte)(sayi-100);
    MessageBox.Show("Sayının 100'den farkı " + sayi.ToString());
}
```

Verilen bu örnekte sayının ilk değeri 50'dir. İşlem sırasında 100 sayısından farkı alınarak sonuç eksi bir değer olur. Eğer, programda sbyte yerine byte değişken türünü kullanmış olsaydık sonuç eksi bir değer olmaz, derleyici rastgele bir sayı üretir ve sonuç yanlış olurdu.

1.1.1.2. Short ve Ushort Değişken Türleri

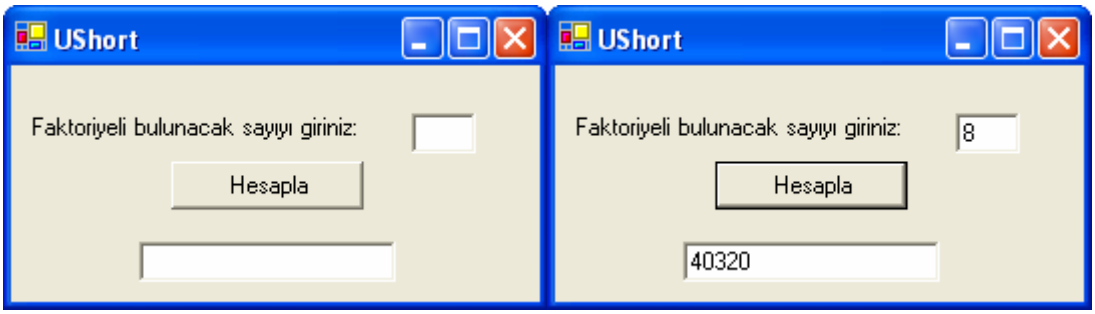
Short ve Ushort değişken türleri 2 byte'lık yer kaplar. Ushort kelimesinin başında bulunan U harfi unsigned (işaretsiz) anlamındadır. Ushort değişken türü, değer aralığındaki pozitif sayıları kapsar.

Örnek

Butona tıklandığında, metin kutusundan girilen bir sayının faktöriyelini hesaplayıp sonucu başka bir metin kutusuna yazdıran program kodu aşağıdadır.

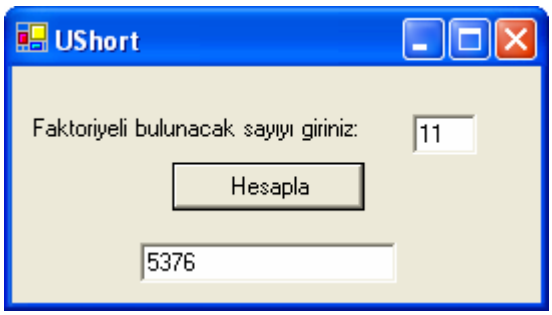
```
private void button1_Click(object sender, System.EventArgs e)
{

    byte sayi,i;
    ushort faktor;
    sayi=Convert.ToByte (textBox1.Text);
    faktor=1;
    for(i=1;i<=sayi;i++)
    {
        faktor=(ushort)(faktor*i);

    }
     textBox2.Text=faktor.ToString();
}
```

Ushort değişken türünü kullanıp faktöriyel hesabı yaptırdığımızda elde edilecek sonuç bu değişken türünün sınırları dahilinde doğru olur (Resim 1.4).



Resim 1.4: UShort değişken tipi üst sınırının gösterimi

Eğer, ushort değişken türünün sınırını aşacak bir değer girilirse hesaplanan sonuç yanlış olur (Resim 1.5). Ancak, derleyici bu yanlış hesaplamadan dolayı bir hata mesajı vermez.



Resim 1.5: UShort değişken tipi sınır aşımının gösterimi

1.1.1.3. Int ve Uint Değişken Türleri

Bu değişken türleri 4 byte'lık veri saklar. İnt ve Uint değişken türleri arasındaki fark, Uint değişken türünün pozitif sayıları, int değişken türünün ise hem pozitif hem de negatif sayıları kapsamasındandır.

Örnek

Int türündeki iki değişkenin ilk değerleri int değişken türünün üst limiti olarak verilip toplama işlemine tabi tutulduğunda elde edilen sonuç int değişken türü değil, uint değişken türünde olur. Buna göre, programda değerleri verilen iki sayının toplamlarının sonucunu mesaj kutusunda (MessageBox) gösteren programın kod satırları aşağıdadır.

```
private void Form1_Load(object sender, System.EventArgs e)
{
    int sayi1,sayi2;
    uint sonuc;
    sayi1=2147483647;
    sayi2=2147483647;
    sonuc=(uint)(sayi1+sayi2);
    MessageBox.Show(sonuc.ToString());
}
```

Program çalıştırıldığında elde edilen sonuç Resim 1.6'daki gibidir.

Resim 1.6: Uint değişken tipi üst sınır gösterimi

Mesaj kutusunda gösterilen bu değer uint değişken türünün alabileceği maksimum değerdir.

1.1.1.4. Long ve Ulong Değişken Türleri

Bu değişken türleri bellekte 8 byte'lık yer kaplar. Aralarındaki fark, Ulong değişken türünün pozitif sayıları içermesidir.

Örnek

Long türündeki iki değişkenin ilk değerleri verilip toplama işlemine tabi tutulduğunda elde edilen sonuç long değişken türü değil, ulong değişken türünde olur. Buna göre, programda değerleri verilen iki sayının toplamının sonucunu mesaj kutusunda gösteren programın kod satırları aşağıdadır.

```
private void Form1_Load(object sender, System.EventArgs e)
{
    long sayi1,sayi2;
    ulong sonuc;
    sayi1=9000000000;
    sayi2=9000000000;
    sonuc=Convert.ToUInt64(sayi1+sayi2);
    MessageBox.Show(sonuc.ToString());
}
```

Program çalıştırıldığında elde edilen sonuç resim 1.7'deki gibi olur.

Resim 1.7: Ulong değişken tipi işlem sonucu

1.1.2. Ondalıklı Değişken Türleri

1.1.2.1. Float Değişken Türü

Ondalıklı sayıları saklamak için kullanılan değişken türüdür. Bellekte 4 byte'lık yer kaplar. Float değişkenlere değer aktarırken değerin sonuna F veya f harfinin yazılması gerekir.

Örnek

Tutar=12500.750f

1.1.2.2. Double Değişken Türü

Bellekte 8 byte'lık yer kaplar. Bu değişken türünde istenirse değerin sonuna D veya d harfi yazılabilir.

Örnek

Toplam=525000.750d

1.1.2.3. Decimal Değişken Türü

Büyük değerleri saklayabilen değişken türüdür. Bellekte 16 byte'lık yer kaplar. Eğer decimal değişken türüne ondalıklı sayı atanmak istenirse değerin sonuna M veya m harfinin yazılması gerekir.

Örnek

Bilanco=78529500000.750 m

1.1.3. Mantıksal Değişken Türü

1.1.3.1. Boolean Değişken Türü

Bellekte 2 byte'lık yer kaplar. Yalnızca true (doğru) ve false (yanlış) değerini alan değişken türüdür. True 1, false 0 rakamına karşılık gelir.

1.1.4. Alfa Sayısal Değişken Türü

1.1.4.1. Char Değişken Türü

Bu değişken türü bellekte 2 byte'lık yer kaplar. Sadece tek karakterlik bilgi için kullanılır. Tek karakterlik bilgi değişkene aktarılırken tek tırnak içinde yazılmalıdır. Char değişkenler harf veya rakam bilgisi saklayabilir.

Char değişken türüne bazı özel görevleri olan tuşları (esc, enter, tab vb.) atamak isterseniz değişken türünün kullanımı şu şekilde olmalıdır.

Değişken_adı = (char) tuş ASCII kodu;
Cevap= (char) 13; Ascii kod tablosunda 13'ün karşılığı Enter tuşudur.

Karakterleri temsil etmek için \ (ters slash) işareti de kullanılır.

\r	enter
\t	tab
\n	satır başı
\e	esc
\\	\

anlamındadır.

Buna göre;
cevap='\r' şeklinde de yazılabilir
.

Char değişken türü Unicode karakter setini de içerdiği için 65536 farklı karakteri içinde tutar.

1.1.4.2. String Değişken Türü

Birden fazla karakter saklamak için kullanılan değişken türüdür. Hem rakamlar hem de harfler için kullanılır. String bilgiler çift tırnak ("") içinde yazılır.

String değişken türüyle yapılan işlemler

Length ( )
Bir stringin karakter uzunluğunu verir.

ToLower ( )
Verilen stringin tüm harflerinin küçük harfe çevrilmesini sağlar. Bu fonksiyon Türkçe karakterleri de küçük harfe dönüştürür.

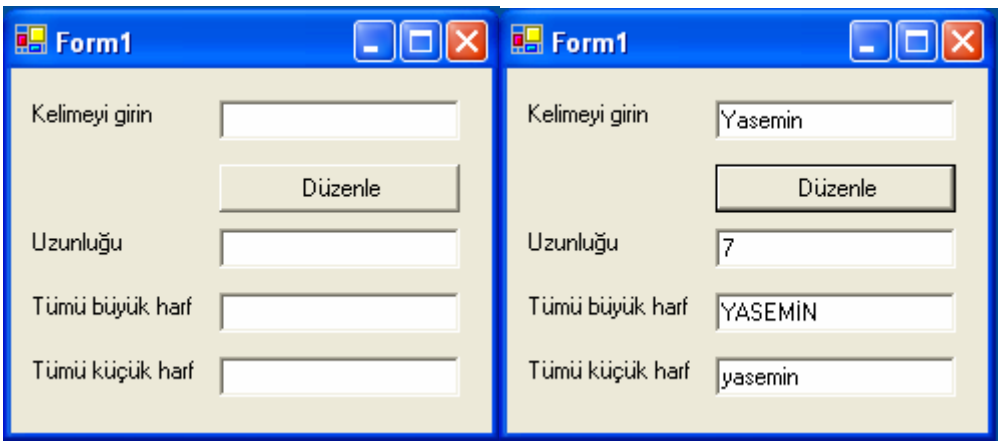
ToUpper ( )
Stringin tüm karakterlerini büyük harfe dönüştürür.

Örnek

Butona tıklandığında, metin kutusundan girilen string bilginin karakter uzunluğunu, tüm karakterlerinin büyük harfe ve küçük harfe çevrilmiş halini ayrı ayrı metin kutularına yazan program kod satırları aşağıdaki gibidir.

```
private void button1_Click(object sender, System.EventArgs e)
{
    textBox2.Text=textBox1.Text.Length.ToString();
    textBox3.Text=textBox1.Text.ToUpper();
    textBox4.Text=textBox1.Text.ToLower();
}
```

Program çalıştırıldığında ekran görüntüsü ve elde edilen sonuçlar resim 1.8'deki gibi olur.



Resim 1.8: Length(), ToUpper() ve ToLower kullanılarak elde edilen ekran görüntüsü

StartWith ( )

Stringin istenilen karakter ya da karakterlerle başlayıp başlamadığını kontrol eder. Sonuç doğru ise True, yanlış ise False değeri geri döndürülür.

Trim ( )

Stringin başında ve sonunda boşluk varsa bu boşlukları atmak için kullanılır. TrimStart stringin başındaki, TrimEnd ise sonundaki boşlukları atar.

SubString ( )

Stringin istenilen bir yerinden istenilen sayıda karakter almak için kullanılır.

IndexOf ( )

Bir stringin içinde başka bir stringin aranmasını sağlar. Aranan string bulunduğunda geriye stringin başlangıç yeri, bulunamadığında ise -1 değeri döndürülür.

Concat ( )
Birden fazla stringi birleştirmek için kullanılır. + operatörü de bu fonksiyonla aynı görevi görür.

Insert ( )
Bir stringe verilen başlangıç yerinden itibaren başka bir stringi eklemeye yarar.

Replace ( )
Bir stringin tamamını veya belirtilen bölümünü başka bir bilgiyle değiştirmek için kullanılır.

Remove ( )
Stringin tamamını veya bir bölümünü silmek için kullanılır.

1.1.5. Object Değişken Türü

Tüm veriler için geçerli olan bir türdür. Bu değişken türüne ondalıklı, string, tam sayı, vb. değişken türleri aktarılabilir.

1.1.6. Pointer Değişkenler

Değişkenlerin bellek adresinden oluşan değişken türüdür. Bellekte 4 byte'lık yer kaplar. Kullanımları güvenli değildir. Çünkü, doğrudan adrese bilgi kaydı yapıldığında kaydedilen yer bir program dosyasının veya sistem dosyasının kayıtlı olduğu adres olabilir. Kullanıldığı durumlarda da sınıfın, fonksiyonun ya da bloğun başına Unsafe yazılması gerekir. Tanımlama sırasında değişken türünün yanına * işareti konur. Bu işaret pointerla belirtilen bellek bölgesinin içeriğini verir. **int* sayi;** örneğinde olduğu gibi.

Bir değişkenin bellekteki adresi elde edilmek istendiğinde & (ampersant) operatörü kullanılır.

1.2. Tür Dönüşümleri

Program içerisinde değişkenlerle ilgili tür dönüşümleri yapmak durumunda kalabilirsiniz. Örneğin, sayısal bir veriyi string değişken türüne, string bir veriyi sayısal değişken türüne dönüştürmeniz gerekebilir. Tür dönüşümlerini gerçekleştirmek için birden fazla seçenek vardır. Nesne tabanlı programlamada bazı tür dönüşümleri derleyici tarafından otomatik olarak yapılırken bazılarının da kullanıcı tarafından yapılması istenir. Herhangi bir değişkenin tür dönüşümü yapılırken dönüştürüleceği değişken türü parantez ( ) içinde yazılıp daha sonra değişkenin ismi yazılır. **(int) y** örneğinde olduğu gibi.

Dönüştürülecek değişken tipinin boyutu hedef değişken tipinin boyutundan büyükse, bu şekildeki bir değişken tipi dönüşümü otomatik yapılamaz. Çünkü işlemin sonucunda byte veya bytelar kaybolacaktır. Derleyici bu riski kabul etmez ve bizden özel komutlar ister. Örneğin, ondalıklı bir sayı tam sayıya yani float tipteki bir sayı int tipteki bir sayıya dönüştürülürken sadece tam kısmı alınacağından ondalıklı kısmı kaybolacaktır ve bu işlem veri kaybına neden olacaktır. Bu şekildeki bir işlemden dolayı derleyici bize hata mesajı vermez. Eğer dönüştürme işlemleri esnasında kullanıcıları bilgilendirmek istersek **checked** deyimini kullanabiliriz.

Derleyici tarafından otomatik olarak tür dönüşümü yapılacak değişken türleri tablo 1.2' de yer almaktadır.

Tür	**Dönüştürülebileceği türler**
byte	short, ushort, int, uint, long, ulong, float, double, decimal
short	int, long, float, double, decimal
int	long, float, double, decimal
long	float, double, decimal
float	double
char	int, uint, long, ulong, float, double, decimal
sbyte	short, int, long, float, double, decimal
ushort	int, uint, long, ulong, float, double, decimal
uint	long, ulong, float, double, decimal
ulong	float, double, decimal

Tablo 1.2: Otomatik dönüşümleri sağlanan değişken türleri ve dönüştürülebildikleri türler

Tür dönüşümü otomatik olarak yapılamayan değişken türleri tablo 1.3' te yer almaktadır.

Tür	**Dönüştürülemeyeceği türler**
byte	sbyte, char
short	sbyte, byte, ushort, uint, ulong, char
int	sbyte, byte, short, ushort, uint, ulong, char
long	sbyte, byte, short, ushort, int, uint, ulong, char
float	sbyte, byte, short, ushort, int, uint, long, ulong, char, decimal
char	sbyte, byte, short
sbyte	byte, ushort, uint, ulong, char
ushort	sbyte, byte, short, char
uint	sbyte, byte, short, ushort, int, char
ulong	sbyte, byte, short, ushort, int, uint, long, char
double	sbyte, byte, short, ushort, int, uint,l ong, ulong, char, float, decimal
decimal	sbyte, byte, short, ushort, int, uint, long, ulong, char, float, double

Tablo 1.3: Otomatik dönüşümleri yapılamayan değişken türleri ve dönüştürülemeyeceği türler

1.2.1. ToString () Metodu

Kod yazım aşamasında işlem yapılan sayısal değerleri string türe dönüştürmeden mesaj kutusu, metin kutusu veya etiketlere aktarmak mümkün değildir. ToString() metodu bu dönüşümü gerçekleştirerek sayısal değerleri string değer alan nesnelere aktarmaktadır.

ToString() metodunu kullanmak için, sayısal değişken yazıldıktan sonra nokta karakteri yazılarak açılan menüden ToString() seçilmelidir. Bu metotla single, int, bool, float ve object türleri string türe dönüştürülebilir.

Örnek

İlk değerleri programda verilen x ve y değişkenlerini form üzerine fareyle tıklandığında metin kutularına yazdıran program kodu aşağıdaki gibidir.

```
private void Form1_Click(object sender, System.EventArgs e)
{
    float x=123.456f;
    long y=98765400;
    textBox1.Text=x.ToString();
    textBox2.Text=y.ToString();
}
```

x ve y değişkenlerinin sonuna .ToString() yazılarak string türe çevrilmiş ve metin kutularına aktarılmıştır.

1.2.2. Convert Metodu

Convert metoduyla dönüştürme işleminde derleyici tarafından izin verilen tüm türlere dönüştürme işlemi yapılabilir. Dönüştürme işlemini yapmadan önce dönüştürülecek bilginin hangi türlere dönüştürülebileceğine dikkat edilmelidir.

Convert metodunu kullanırken; Convert yazıp nokta karakteri yazıldığında açılan menüden istenilen dönüşüm türü seçilerek dönüştürülecek bilgi parantez içinde yazılmalıdır.

Örnek

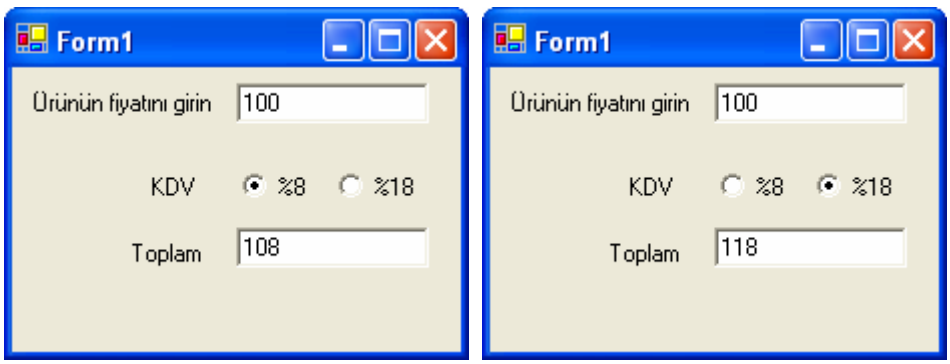
Convert.ToString(textBox5.Text);

Örnek

Metin kutusundan girilen ürün fiyatına göre KDV oranını %8 ve %18 olarak radyo düğmelerinden yapılan seçime göre hesaplayan ve sonucunu yine bir metin kutusuna yazan programın kod satırları aşağıdaki gibidir.

```
private void radioButton1_CheckedChanged(object sender, System.EventArgs e)
{
    Single fiyat;
    fiyat=Convert.ToSingle(textBox1.Text);
    textBox2.Text=Convert.ToString(fiyat+(fiyat*0.08));
}
private void radioButton2_CheckedChanged(object sender, System.EventArgs e)
{
    Single fiyat;
    fiyat=Convert.ToSingle(textBox1.Text);
    textBox2.Text=Convert.ToString(fiyat+(fiyat*0.18));
}
```

Program kodunda Convert metodunun ToSingle() ve ToString() metodu kullanılmıştır. İşlem sonucunda da resim 1.9'daki ekran çıktıları elde edilmiştir.



Resim 1.9: Convert metodunun ToSingle() ve ToString() metotlarının kullanılmasıyla elde edilen ekran görüntüsü

Convert metodu kullanılırken bazı dönüşümler nokta karakterinde sonra açılan menüde farklı şekilde karşınıza çıkar. Örneğin, ToShort metodu diye bir metot yoktur. Nesne tabanlı programımızda bunun karşılığı ToInt16'dır. Menüde olmayan metotlar için tablo 1.4'teki metotları kullanabilirsiniz.

Tür adı	Karşılığı
short	Convert.ToInt16
int	Convert.ToInt32
long	Convert.ToInt64
ushort	Convert.ToUInt16
uint	Convert.ToUInt32
ulong	Convert.ToUInt64

Tablo 1.4: Tür adları ve karşılıkları

1.2.3. Parse Metodu

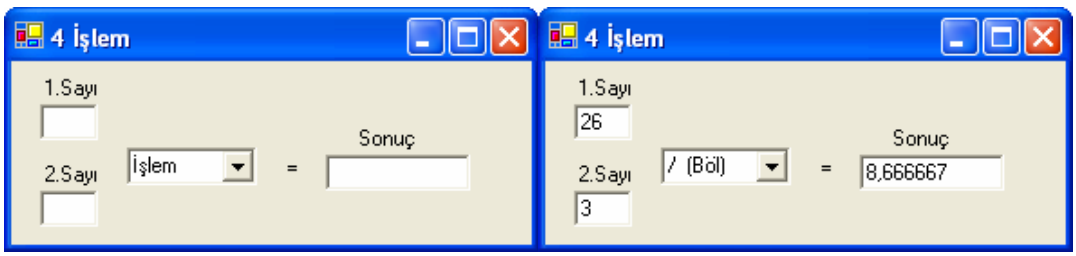
Tür dönüşümleri için kullanılan yöntemlerden biri de parse metodudur. Bu metot da Convert metodunun yaptığı ve izin verilen tüm dönüşümleri yapmaktadır.

Örnek

Ayrı ayrı metin kutularına girilen iki sayıyla açılır liste kutusundan (comboBox) yapılan seçime göre işlem yapıp sonucu yine bir metin kutusuna yazan program kodu aşağıdaki gibidir.

```
private void comboBox1_SelectedIndexChanged(object sender, System.EventArgs e)
{
    int sayi1,sayi2;
    if(comboBox1.SelectedIndex==0)
    {
        sayi1=int.Parse(textBox1.Text);
        sayi2=int.Parse(textBox2.Text);
        textBox3.Text=Convert.ToString(sayi1+sayi2);
    }
    if(comboBox1.SelectedIndex==1)
    {
        sayi1=int.Parse(textBox1.Text);
        sayi2=int.Parse(textBox2.Text);
        textBox3.Text=Convert.ToString(sayi1-sayi2);
    }
    if(comboBox1.SelectedIndex==2)
    {
        sayi1=int.Parse(textBox1.Text);
        sayi2=int.Parse(textBox2.Text);
        textBox3.Text=Convert.ToString(sayi1*sayi2);
    }
    if(comboBox1.SelectedIndex==3)
    {
        sayi1=int.Parse(textBox1.Text);
        sayi2=int.Parse(textBox2.Text);
        float bolum;
        bolum=(float)sayi1/sayi2;
        textBox3.Text=Convert.ToString(bolum);
    }
}
```

Bu örnekte metin kutularından girilen sayılar string değerde olduğu için int.Parse ile sayısal türe çevrilerek aritmetiksel işlemler yapılmıştır. İşlem sonucunda elde edilen ekran görüntüsü resim 1.10'daki gibidir.



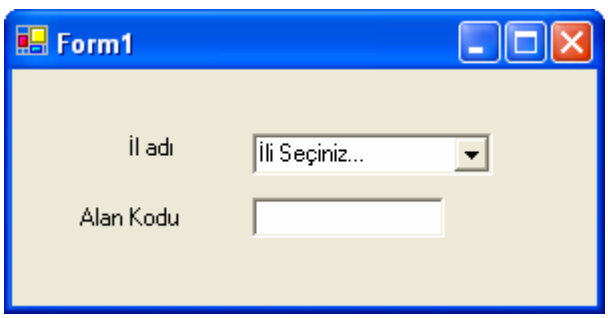
Resim 1.10: Parse metoduyla tür dönüşümü yapılan programın ekran görüntüsü

1.3. Enum Yapısı

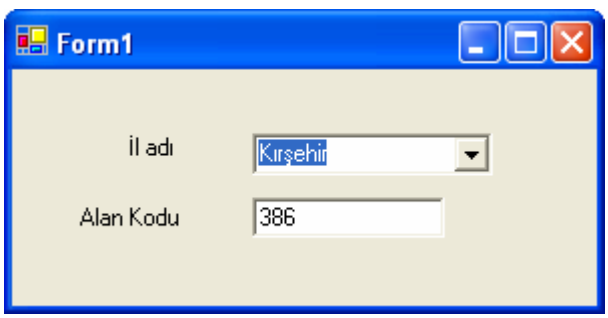
Enumerations (Enum), sayılabilir tipler için kullanılan numaralandırıcılardır. Enum yapısı kullanıcı tanımlı bir yapıdır. Enum yapısının elemanları 0'dan başlar ve index değerleri birer birer artar. Dolayısıyla numaralandırıcıya ait elemana ulaştığımızda bu elemanın index değerine de sahip oluruz.

Örnek

Açılır liste kutusundan seçilen il adına göre telefon alan kodunu metin kutusuna yazan programın tasarlanmış ekran görüntüsü resim 1.11'de, yapılan işlem sonucunun ekran görüntüsü resim 1.12'de gösterilmiştir..



Resim 1.11: Formun tasarlanması



Resim 1.12: İşlemin görüntülenmesi

Bu ekran görüntüsünü elde edebilmek için yazılan program kodu da aşağıdaki gibidir.

```
private void comboBox1_SelectedIndexChanged(object sender, System.EventArgs e)
{   short kod;
    if(comboBox1.Text=="Ankara")
    {   kod=(short)alankodu.Ankara ;
        textBox1.Text=kod.ToString();
    }
    if(comboBox1.Text=="İzmir")
    {   kod=(short)alankodu.İzmir ;
        textBox1.Text=kod.ToString();
    }
    if(comboBox1.Text=="Muğla")
    {   kod=(short)alankodu.Muğla ;
        textBox1.Text=kod.ToString();
    }
    if(comboBox1.Text=="Kırşehir")
    {   kod=(short)alankodu.Kırşehir ;
        textBox1.Text=kod.ToString();
    }
    if(comboBox1.Text=="Antalya")
    {   kod=(short)alankodu.Antalya ;
        textBox1.Text=kod.ToString();
    }
}
private void Form1_Load(object sender, System.EventArgs e)
{
    string [] dizi=alankodu.GetNames(typeof(alankodu));
    for(int i=0;i<5;i++)
        comboBox1.Items.Add(dizi[i]);
}
enum alankodu:short
{
    Ankara=312,
    İzmir=232,
    Muğla=252,
    Kırşehir=386,
    Antalya=242
}
```

1.4. Struct (Yapı)

Birden fazla, farklı türdeki değişkenlerin tanımlandığı yapıdır. Birbiriyle ilişkili olan değişkenler struct yapıyla tanımlanır. Dolayısıyla kullanıcı kendi değişken tipini oluşturur. Oluşturduğu bu değişkene bir isim vermelidir.

```
struct  personel
{
  public string adı;
  public string soyadı;
  public string adres;
}
```

personel kisi= new personel();

yapının elemanlarına bilgi aktarmak için;
kisi.adı=”Ahmet”;
kisi.soyadı=”Can”;
kisi.adres=”ANKARA”;

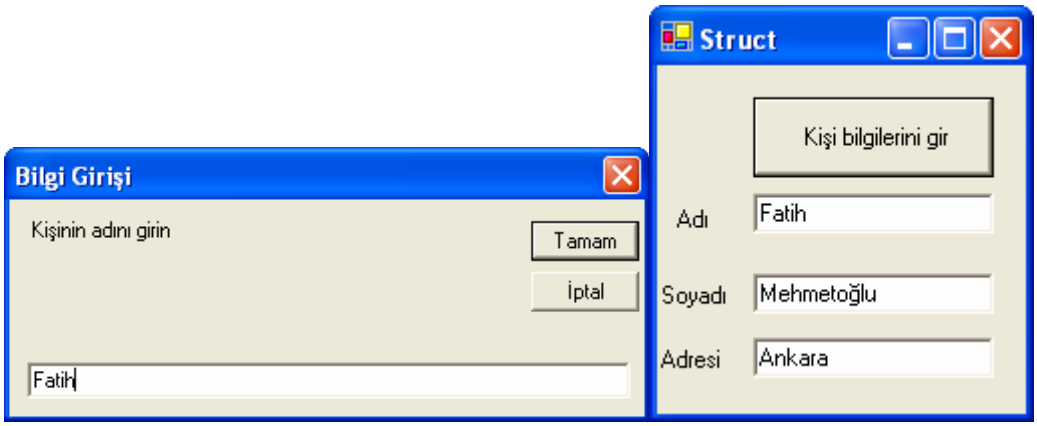
Tanımlanan yapı, tüm programda yani başka class ve formlarda kullanılmak istenirse **public** olarak tanımlanmalıdır.

Örnek

Struct bir yapı oluşturularak kişinin ad, soyad ve adres bilgilerini InputBox()’tan alıp metin kutularına yazdıran program kodu aşağıda ve ekran görüntüleri resim 1.13’teki gibidir.

```
public struct personel
{
    public string ad;
    public string soyad;
    public string adres;
}
public personel kisi=new personel();
private void button1_Click(object sender, System.EventArgs e)
{
kisi.ad=(Microsoft.VisualBasic.Interaction.InputBox("Kişinin adını girin"
    ,"Bilgi Girişi","",20,20)).ToString();
kisi.soyad=(Microsoft.VisualBasic.Interaction.InputBox("Kişinin soyadını girin"
    ,"Bilgi Girişi","",20,20)).ToString();
kisi.adres=(Microsoft.VisualBasic.Interaction.InputBox("Kişinin adresini girin"
    ,"Bilgi Girişi","",20,20)).ToString();
textBox1.Text=kisi.ad;
    textBox2.Text=kisi.soyad;
    textBox3.Text=kisi.adres;
}
```

Not : Bu nesne tabanlı programlama dilinde diğer dillerin özelliklerini de kullanmak mümkündür. Visual Basic programlama dilinin InputBox() metodunun bu programda kullanılması için projenize eklemeniz gerekir. Bunun için, Project menüsünden Add Reference… komutu seçilerek açılan Add Reference iletişim penceresinden Microsoft Visual Basic.NET Runtime dosyasını seçip OK düğmesine tıklayınız. Artık InputBox() metodunu projenizde kullanabilirsiniz.



Resim 1.13: Struct yapı kullanılarak elde edilen sonuçların görüntülenmesi

UYGULAMA FAALİYETİ-1

İşlem Basamakları	Öneriler
➢ İki ayrı metin kutusundan iki farklı sayı giriniz. Bu sayıların değişken türlerini sınırlarına göre seçiniz.	➢ Değişken ismi olarak a ve b harflerini kullanabilirsiniz.
➢ Üçüncü bir değişken daha tanımlayınız.	➢ Bu değişkenin değişken türünü yapılacak işlemin sonucuna göre belirleyebilirsiniz.
➢ Tanımladığınız ilk iki değişkeni birbirine bölerek üçüncü değişkene aktarınız.	➢ Bölme işlemi yapılacağından üçüncü değişkenin değişken türü ondalıklı değişken türü olarak belirlenmelidir.
➢ Ekrana iki değişkeni ve işlem sonucunu gösteren değişkeni mesaj kutusunda yazdırınız.	➢ Hangi sayılarla hangi işlemin yapıldığını göstermek için bütün verileri gösterebilirsiniz ya da sadece sonuç değerini ekranda gösterebilirsiniz.
➢ Şimdiye kadar yazdığınız kod satırlarını gözden geçiriniz.	➢ Kod satırlarını yazarken dikkatli olunuz. Programa dillerinde her bir işaretin önemi çok büyüktür.
➢ Programı çalıştırınız. Çalışma esnasında hata oluşmuşsa kod satırlarına dönerek yazım hatalarınızı kontrol edip tekrar çalıştırınız.	➢ Amacınızı, kod satırlarını ve işlem sonucunun ekran görüntüsünü defterinize yazınız.

UYGULAMA FAALİYETİ-2

Aşağıda verilen soruları ödev olarak yapınız. Sonuçları rapor şeklinde öğretmeninize sununuz.

- Metin kutularından girilen sayılardan ikincisinin 0'dan farklı olması durumunda iki sayıyı toplayıp bir başka metin kutusuna yazdıran programın kod satırlarını ve ekran görüntüsünü hazırlayınız.

- Başlangıçta sabit olarak verilen değerlerle metin kutusundan girilen değerin eşit olup olmadığına bakarak ne tür bir üçgen olduğunu bulup mesaj kutusuyla ekranda gösteren programın kod satırlarını ve ekran görüntüsünü hazırlayınız.

Metin kutularından girilen üç sayıdan en küçük olanı buldurup mesaj kutusuyla ekranda gösteren programın kod satırlarını ve ekran görüntüsünü hazırlayınız

ÖLÇME VE DEĞERLENDİRME

A. OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki soruları dikkatlice okuyarak doğru/yanlış seçenekli sorularda uygun harfleri yuvarlak içine alınız. Seçenekli sorularda ise uygun şıkkı işaretleyiniz.

1. Değişkenler bellekte (RAM) sürekli kalır.(D/Y)

2. Sabitler program içinde istenildiği zaman değiştirilir. (D/Y)

3. Nesne tabanlı programlama diline özel, sadece bu dilde kullanılan değişken aşağıdakilerden hangisidir?
   A) byte
   B) int
   C) uint
   D) decimal

4. Sbyte değişken türünün değer aralığında yer almayan sayı hangisidir?
   A) -128
   B) 127
   C) 0
   D) -127

5. Char değişken türü unicode özelliğinden dolayı 65536 farklı karakter tutabilir. (D/Y)

6. Tür dönüşümleri esnasında veri kaybı söz konusu değildir. (D/Y)

7. Ushort değişken türü hem negatif hem de pozitif değer alır. (D/Y)

8. Aşağıdaki harflerden hangisi değişken türlerini sembolize eden ve değerin yanına yazılan harflerden biri değildir?
   A) L
   B) M
   C) F
   D) D

9. Pointer değişkenlerin kullanılması programa hız kazandırmasına rağmen pek güvenli değildir. (D/Y)

10. Enum yapısını kullanarak programcı kendine göre değişkenler tanımlayabilir. (D/Y)

DEĞERLENDİRME

Cevaplarınızı cevap anahtarı ile karşılaştırınız. Doğru cevap sayınızı belirleyerek kendinizi değerlendiriniz. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt yaşadığınız sorularla ilgili konulara geri dönerek tekrar inceleyiniz. Tüm sorulara doğru cevap verdiyseniz diğer öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ-2

AMAÇ

Dizi mantığını ve dizi değişkenlerini program içerisinde uygun şekilde tanımlayıp kullanabileceksiniz.

ARAŞTIRMA

- Bir kitapçıya gittiğinizde çeşitli türlerde bulunan kitaplar neden ayrı konu başlıklarıyla tasnif edilmişlerdir? Araştırınız.
- Bir değişken aynı anda kaç değer saklayabilir? Araştırınız.

2. DİZİLER

2.1. Dizi Tanımlama

Bellekte aynı türden sıralanmış verilerin oluşturduğu yapıya **dizi** denir. Tek bir tanımlama yapılarak istenilen sayıda veri girişi sağlanabilir. Dizi elemanları bir indis numarasına sahiptir. İndis numaraları varsayılan olarak sıfırdan başlar. Dizinin kaç elemanlı olacağı dizinin tanımlandığı satırda veya daha sonra da belirtilebilir. Ayrıca dizi tanımlaması yapıldığında dizinin eleman sayısı, kullanılacak eleman sayısından az ise dizi yeniden boyutlandırılabilir.

Dizi değişkenin eleman sayısını belirtmek için **new** metodu kullanılır.

Örnek

5 elemanlı bir rakam dizisi tanımlanacak olursa;

```
int [] rakam;
rakam= new int[5];          komut satırları yazılarak tanımlama yapılmış olur.
```

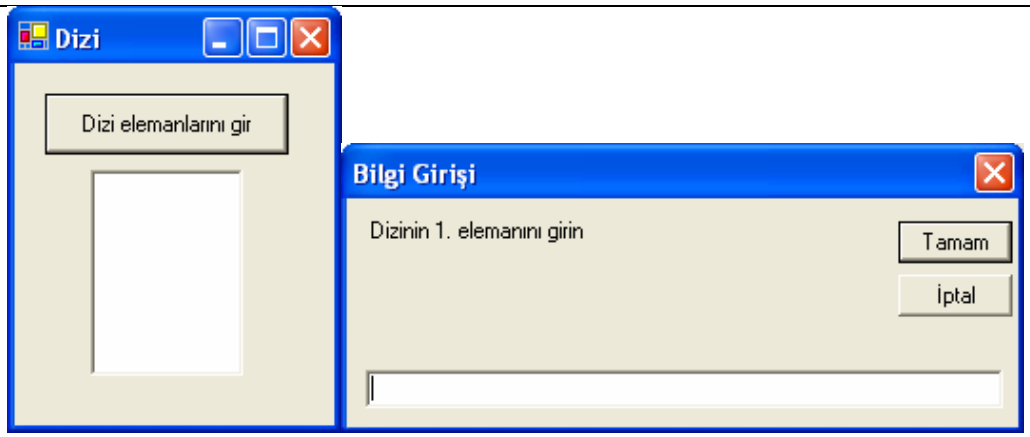
Ayrıca,
int[] Rakam= new int[5]; komut satırı yazılarak tek satırda da tanımlama yapılabilir.

Bu şekilde tanımlanan bir dizinin ilk elemanının indis numarası 0 ve son elemanının indis numarası 4'tür.

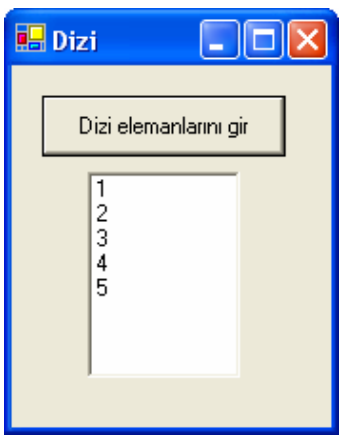
Örnek

Butona tıklandığında, 5 elemanlı bir int dizi tanımlayarak dizi elemanlarını InputBox'tan okutan ve bu elemanları liste kutusunda listeleyen program kodu aşağıda ve ekran görüntüleri resim 2.1 ve resim 2.2'deki gibidir.

```
private void button1_Click(object sender, System.EventArgs e)
{   byte i;
    int[] dizi1= new int[5];
    for(i=0;i<5;i++)
    {
        dizi1[i]=int.Parse(Microsoft.VisualBasic.Interaction.InputBox("Dizinin "
            + (i+1)+". elemanını girin","Bilgi Girişi","",20,20));
    }
    for (i=0;i<5;i++)
    {
        listBox1.Items.Add(dizi1[i]);
    }
}
```



Resim 2.1 : Dizi elemanlarının bilgi giriş kutusundan girilmesi

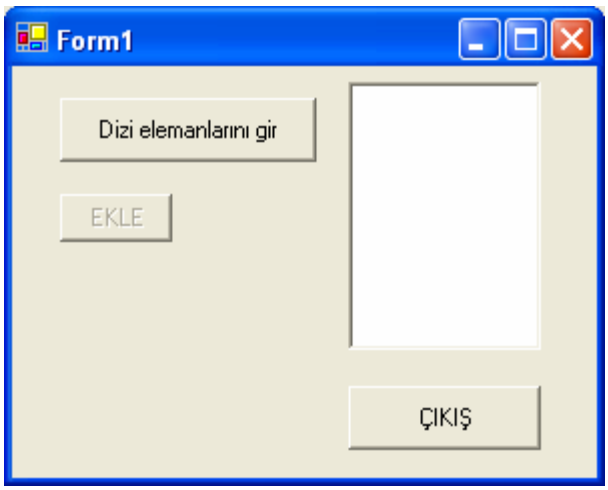


Resim 2.2: Bilgi girişinden sonra elemanların liste kutusunda listelenmesi

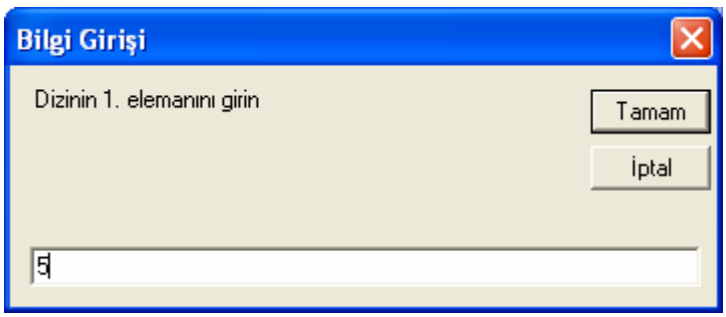
Örnek

5 elemanlı bir int dizi tanımlanarak, dizi elemanları "Dizi elemanlarını gir" düğmesine tıklandığında InputBox() ile diziye diziden de liste kutusuna eklenmektedir. Başlangıçta "EKLE" düğmesi pasif haldedir. Dizi elemanlarının girişinden sonra "Dizi elemanlarını gir" düğmesi pasif hale "EKLE" düğmesi aktif hale gelmektedir. Ekle düğmesine tıklandığında dizi yeniden boyutlandırılarak InputBox()'tan diziye yeni eleman girişi yapılmakta ve aynı zamanda liste kutusuna da eklenmektedir. Çıkış düğmesiyle de program sonlandırılmaktadır. Programın kod satırları aşağıda ekran görüntüleri resim 2.3, 2.4, 2.5, 2.6 ve 2.7'deki gibidir.

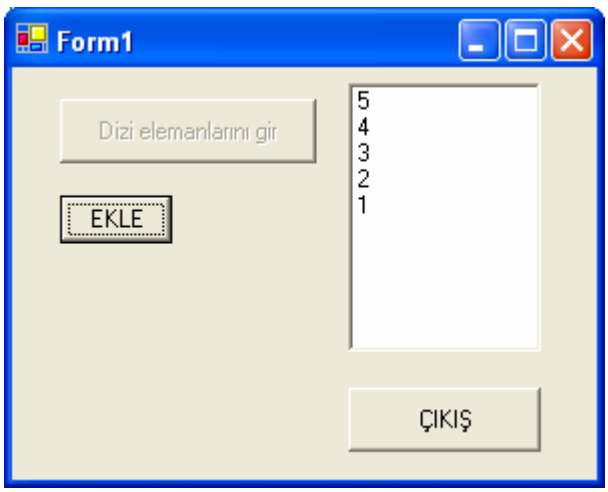
```
int sayi;
private void button1_Click(object sender, System.EventArgs e)
{   byte i;
    int[] dizi1= new int[5];
    for(i=0;i<5;i++)
    {
    dizi1[i]=int.Parse(Microsoft.VisualBasic.Interaction.InputBox("Dizinin "
        + (i+1)+". elemanını girin","Bilgi Girişi","",20,20));
    }
    for (i=0;i<5;i++)
    {
        listBox1.Items.Add(dizi1[i]);
    }
    sayi=5;
    button1.Enabled=false;button2.Enabled=true;
}
byte s=5;
private void button2_Click(object sender, System.EventArgs e)
{   int[] dizi1=new int[sayi+1];
dizi1[sayi]=int.Parse(Microsoft.VisualBasic.Interaction.InputBox("Dizinin "
    + (s+1)+". elemanını girin","Bilgi Girişi","",20,20));
    listBox1.Items.Add(dizi1[sayi]);
    s++;
}
private void button3_Click(object sender, System.EventArgs e)
{
    Close();
}

private void Form1_Load(object sender, System.EventArgs e)
{
    button2.Enabled=false;
}
```



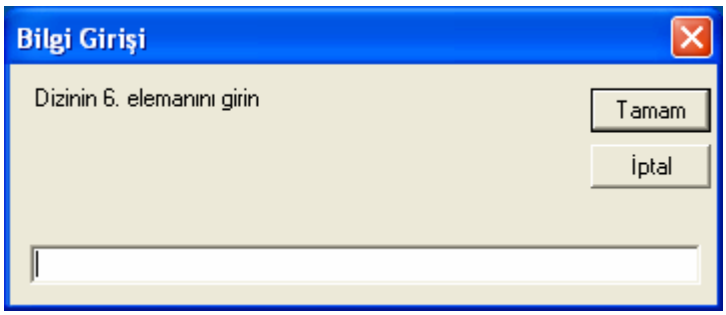
Resim 2.3:Dizi elemanlarını gir düğmesiyle diziye bilgi girişi yapılmaktadır.



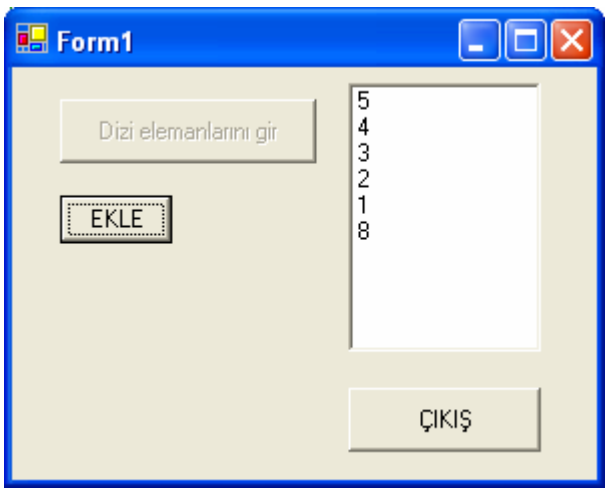
Resim 2.4: Diziye bilgi girişi InputBox() ile sağlanmaktadır.



Resim 2.5: Diziye eleman girişi bittikten sonra EKLE düğmesi aktif hale gelmektedir.



Resim 2.6 : EKLE düğmesine tıklandığında tekrar bilgi girişi sağlanır.



Resim 2.7 : Dizi yeniden boyutlandırılarak InputBox()'tan girilen yeni değer hem diziye hem de liste kutusuna eklenir.

2.1.1. Dizilere İlk Değer Atama

Dizinin tanımlanması sırasında ilk değerleri de beraberinde verilebilir. Bir dizinin elemanlarına farklı değişken türlerinde bilgiler aktarılmak istenirse dizi değişkenin türü **object** olmalıdır.

Örnek
Tanımlanan rakam dizisinin elemanlarının ilk değerleri aşağıda verilmiştir.
Rakam [0]= 2;
Rakam [1]= 3;
Rakam [2]= 7;
Rakam [3]= 5;
Rakam [4]= 9;

Örnek

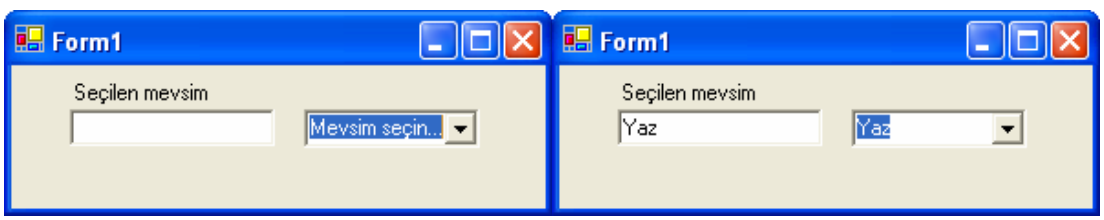
Mevsimler string dizisinin tanımlanması ve ilk değerlerinin verilmesi şu şekildedir.

string[] Mevsimler =new.string[4] {“İlkbahar”,”Yaz”,”Sonbahar”,”Kış”};

Örnek

4 elemanlı string bir dizi tanımlanıp ilk değerleri verilerek formun çalıştırılmasında bu değerler comboBox() ta listelenmektedir. Açılır liste kutusundan (comboBox) yapılan seçimi metin kutusunda gösteren program kodu aşağıda ve ekran görüntüleri resim 2.8‘dedir.

```
public string[] mevsimler=new string[4]{"İlkbahar","Yaz","Sonbahar","Kış"};
private void Form1_Load(object sender, System.EventArgs e)
{
        comboBox1.Items.Add(mevsimler[0]);
        comboBox1.Items.Add(mevsimler[1]);
        comboBox1.Items.Add(mevsimler[2]);
        comboBox1.Items.Add(mevsimler[3]);
}

private void comboBox1_SelectedIndexChanged(object sender, System.EventArgs e)
{
    if (comboBox1.SelectedIndex==0)
                    textBox1.Text=mevsimler[0];
    if (comboBox1.SelectedIndex==1)
        textBox1.Text=mevsimler[1];
    if (comboBox1.SelectedIndex==2)
        textBox1.Text=mevsimler[2];
    if (comboBox1.SelectedIndex==3)
        textBox1.Text=mevsimler[3];
}
```



Resim 2.8: Açılır listeden yapılan seçimin metin kutusunda gösterilmesi

Ayrıca, ilk değer ataması yapılacak olan dizilerde eleman sayısını belirtmeye gerek yoktur.

string [] Mevsimler= {“İlkbahar”,”Yaz”,”Sonbahar”,”Kış”};

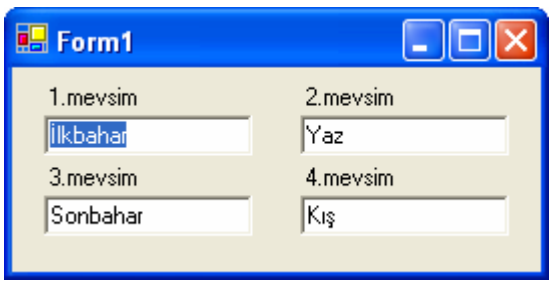
Bu tanımlamaya göre new metodunu kullanmayabilirsiniz. Çünkü dizi değişkenin eleman sayısı verilen ilk değerlerin sayısına göre belirlenir.

Örnek

Boyut belirtmeden tanımlanan dizi elemanlarını metin kutularına yazdıran program kodu şu şekildedir.

```
public string[] mevsimler={"İlkbahar","Yaz","Sonbahar","Kış");
private void Form1_Load(object sender, System.EventArgs e)
{
    textBox1.Text=mevsimler[0];
    textBox2.Text=mevsimler[1];
    textBox3.Text=mevsimler[2];
    textBox4.Text=mevsimler[3];
}
```

Böylece uygulamanın ekran görüntüsü resim 2.9'daki gibi olur.



Resim 2.9 : Dizi elemanlarının metin kutularına yazdırılması

2.2. Çok Boyutlu Diziler

İki veya daha çok boyutlu bilgileri saklamak için kullanılan dizi türleridir. Tek boyutlu dizilerde olduğu gibi çok boyutlu dizilerde de ilk değer ataması yapılabilir. Çok boyutlu dizilerde işlem yapabilmek için genellikle iç içe döngü yapıları kullanılmaktadır.

Örnek

int [,] dizi1;
dizi1 = new int [3,2]; şeklinde tanımlama yapılır.

[,] şeklindeki bu gösterim dizinin iki boyutlu olduğunu belirtir. (,) virgül sayısı arttıkça dizinin boyutu da artar.

Örnek

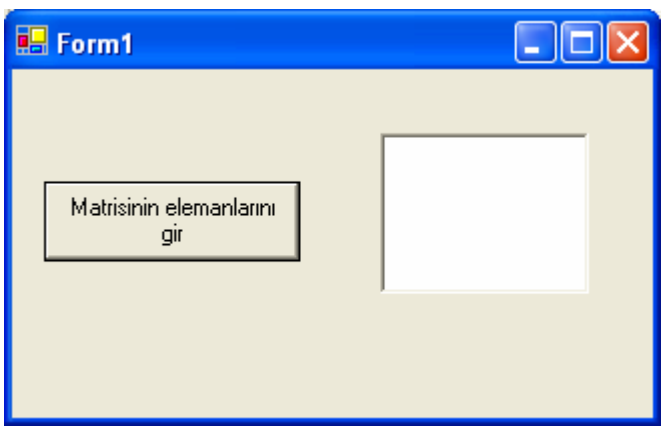
2x2'lik bir matrise ilk değer ataması aşağıdaki gibi yapılabilir.
int [,] Matris={{1,0},{0,1}};

Bir dizinin boyutu öğrenilmek istenirse **Rank ()** metodu kullanılır.

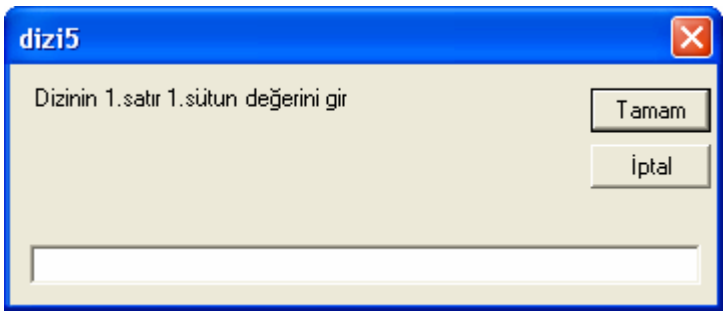
Örnek

"Matrisin elemanlarını gir" düğmesine tıklandığında 3x2'lik bir matrise eleman girişi InputBox() ile yapılmaktadır. Dizi giriş işlemi bittikten sonra girilen elemanlar "Multiline" özelliğine sahip metin kutusuna girilerek satır-sütun halinde yazdıran program kodu aşağıdaki gibi olur.

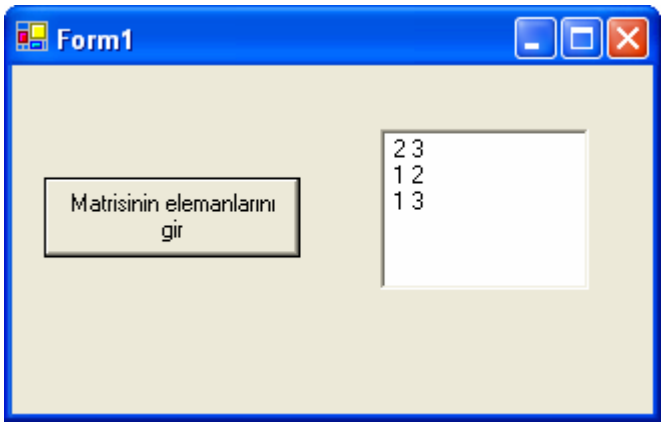
```
private void button1_Click(object sender, System.EventArgs e)
{
int [,] matris=new int[3,2];
for(int i=0;i<3;i++)
  for(int j=0;j<2;j++)
    matris[i,j]=int.Parse(Interaction.InputBox("Dizinin " + (i+1)+
        ".satır " + (j+1)+".sütun değerini gir","","",20,20));
for(int i=0;i<3;i++)
{
    for(int j=0;j<2;j++)
    {
    textBox1.Text=textBox1.Text + " " +matris[i,j].ToString();
    }
textBox1.Text=textBox1.Text + "\r\n";
}
}
```



Resim 2.10: Formun ilk çalışma hali



Resim 2.11: Dizi elemanlarının InputBox() ile girilmesi



Resim 2.12: Dizi elemanlarının girişi tamamlandıktan sonra multiline özelliğine sahip metin kutusunda satır-sütun halinde gösterilmesi

2.3. Karışık (Düzensiz) Diziler

Bu dizilere **Jagged** diziler de denir. Karışık diziler sütun sayısı birbirinden farklı olan dizilerdir. Bir başka deyişle, farklı sütun sayılarına ait diziler dizisidir.

Örnek

```
int [ ] [ ] dizi = new int [3][ ];
```

Bu tanıma göre 1. boyut 3 elemanlı olarak tanımlanmışken diğer boyutun kaç olacağı belli değildir. Tanımlanmayan diğer boyutun eleman sayısı ilk değer ataması sonucunda belirlenir. Buna göre;

```
dizi[0]= new int [4];
dizi[1]= new int [5];
dizi[2]= new int [3];
dizi[3]= new int [2];
```

şeklindeki tanımlamayla dizinin 2. boyutunun ilk satırının eleman sayısı 4'tür. İstenirse dizinin 2. boyutu çok boyutlu bir dizi olabilir.

Örnek

1. boyutu 3 olan diğer boyutu belli olmayan bir jagged dizi tanımlanarak 2. boyutunun elemanlarını form yüklendiğinde diziye girilmektedir. Her bir sütun elemanlarını ayrı ayrı liste kutusuna listeleyen ve bu dizinin rankını "Dizinin rankını göster" düğmesine basıldığında mesaj kutusunda gösteren program kodu aşağıdaki gibidir.

```
// 3 elemanlı bir dizi tanımlanır
string[][] aile = new string[3][];
private void Form1_Load(object sender, System.EventArgs e)
{
    // İlk dizinin 4 elemanlı başlangıç değerleri verilir.
    aile[0] = new string[] {"Can", "Anne", "Baba", "Haluk amca");
    // İkinci dizinin 5 elemanlı başlangıç değerleri verilir.
    aile[1] = new string[] {"Alican", "Anne", "Baba", "Ayşe", "Küçük veli");
    // Üçüncü dizinin 3 elemanlı başlangıç değerleri verilir.
    aile[2] = new string[] {"Bartu", "Selçuk", "Veli");

    for (int j=0;j<4;j++)
    listBox1.Items.Add(aile[0][j]);
    for (int j=0;j<5;j++)
    listBox2.Items.Add(aile[1][j]);
    for (int j=0;j<3;j++)
        listBox3.Items.Add(aile[2][j]);
}
private void button1_Click_1(object sender, System.EventArgs e)
{
    MessageBox.Show("Aile dizisi "+(aile.Rank).ToString()+" boyutludur","Rank");
}
```

Resim 2.13: Liste kutularına listelenen karışık dizi ve rankının gösterilmesi

2.4. Koleksiyonlar

Dizi eleman sayısının esnek şekilde tanımlanması için kullanılan yapılardır. Eleman sayısı, programda kullanılacak eleman sayısına göre artırılır ya da azaltılır.

Koleksiyonlar, **System.Collections** isim alanında (namespace) bulunur. Koleksiyonlarla çalışmaya başlamadan önce System.Collection'ı using deyimiyle projenin başına dâhil etmek gerekir.

2.4.1. ArrayList Sınıfı

Dinamik olarak büyüyüp küçülebilen yani çalışma esnasında eleman sayısı değiştirilebilen diziler için kullanılır. Diğer dizilerden farklı olarak dizinin elemanları aynı değişken türünden olmak zorunda değildir.

Örnek

ArrayList kitaplar= new ArrayList( );

kod satırıyla kitaplar adlı esnek yapılı bir dizi tanımlanmıştır.

Dizi içindeki bir elemanı bulmak için [ ] operatörünün içine index numarası yazılıp eleman bulunabilir.

ArrayList sınıfıyla bir dizi oluşturulduğunda dizinin eleman sayısı belirtilmemişse varsayılan değeri 16'dır. Eleman sayısı 16'yı geçerse dizinin eleman sayısı otomatik olarak 32'ye yükseltilir.

2.4.1.1. Add Metodu()

ArrayList dizisine eleman eklemek için kullanılır. Eklenen eleman dizinin en sonuna eklenir. ArrayList koleksiyonuyla bir dizi tanımlanırken mutlaka ilk değer verilmelidir. Çünkü eklenen bu eleman, dizinin 0. indexini oluşturur (Zero-Based). Daha sonra eklenen her eleman bir sonraki index numarasına sahip olur.

Örnek

ArrayList sınıfıyla kitaplar adında bir dizi oluşturarak Add() metoduyla 4 eleman diziye aktarılmıştır. Aktarılan elemanlardan indis numarası 3 olan dizi elemanın mesaj kutusunda gösteren program kodu aşağıda ve ekran görüntüsü de resim 2.14'te gösterilmiştir.

```
private void Form1_Load(object sender, System.EventArgs e)
{
    ArrayList kitaplar=new ArrayList ();
    kitaplar.Add("Bilgisayar");
    kitaplar.Add("Bilimsel");
    kitaplar.Add("Tarih");
    kitaplar.Add("Sanat");
    MessageBox.Show(kitaplar[3].ToString(),"Dizi elemanı");
}
```

Resim 2.14: Belirtilen elemanın mesaj kutusunda gösterimi

2.4.1.2. Capacity Özelliği

ArrayList dizisinin mevcut boyutu öğrenilmek istendiğinde kullanılan özelliktir.

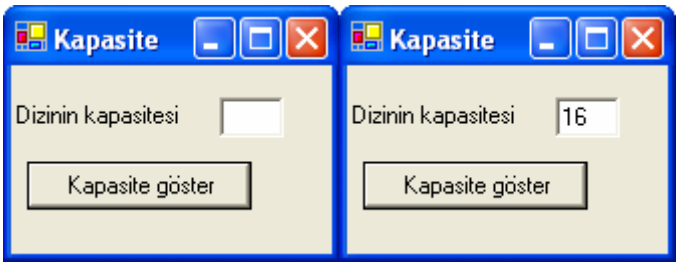
MessageBox (kitaplar.Capacity);

Örnek

ArrayList ile oluşturulan ve elemanları başlangıçta diziye aktarılan kitaplar dizisinin kapasitesi, butona tıklandığında metin kutusuna yazdırılmaktadır.

```
ArrayList kitaplar=new ArrayList ();
private void Form1_Load(object sender, System.EventArgs e)
{

    kitaplar.Add("Bilgisayar");
    kitaplar.Add("Bilimsel");
    kitaplar.Add("Tarih");
    kitaplar.Add("Sanat");
}

private void button1_Click(object sender, System.EventArgs e)
{
textBox1.Text=kitaplar.Capacity.ToString();
}
```

Program çalıştırıldığında ve "Kapasite göster" düğmesine tıklandığında ekran görüntüleri resim 2.15'te gösterildiği gibidir.



Resim 2.15: ArrayList dizisinin kapasitesi

2.4.1.3. Insert () Metodu

ArrayList dizisinin belli bir konumuna eleman eklemek için kullanılır.

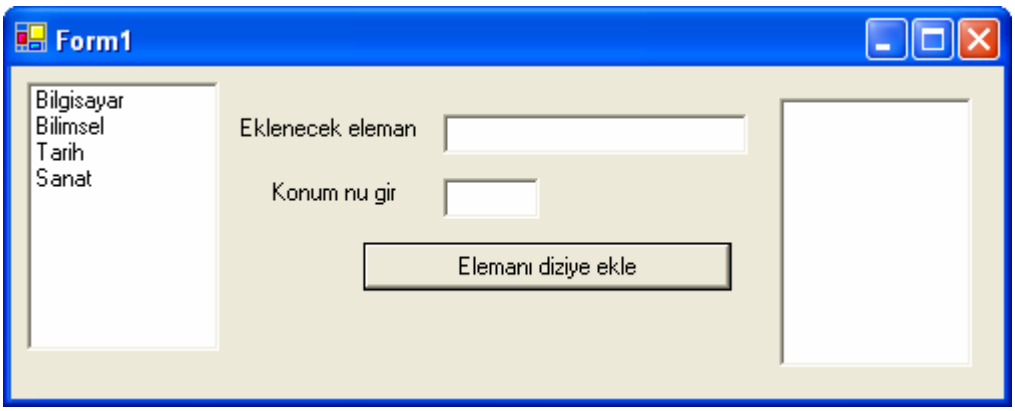
Kullanımı, dizi.Insert (no,eleman); şeklindedir.

Örnek

ArrayList sınıfıyla public olarak oluşturulan kitaplar dizisinin elemanları, form yüklendiğinde diziye aktarılmakta ve aktarılan elemanlar da bir liste kutusunda listelenmektedir. "Elemanı diziye ekle" düğmesine tıklandığında birinci metin kutusundan girilen eleman adıyla, ikinci metin kutusundan girilen, elemanın dizide hangi konuma ekleneceği belirten numarayla, Insert() metodu kullanılarak kitaplar dizisine eklenmektedir. Dizinin son halini bir başka liste kutusunda listeleyen programın kod satırları şu şekildedir.

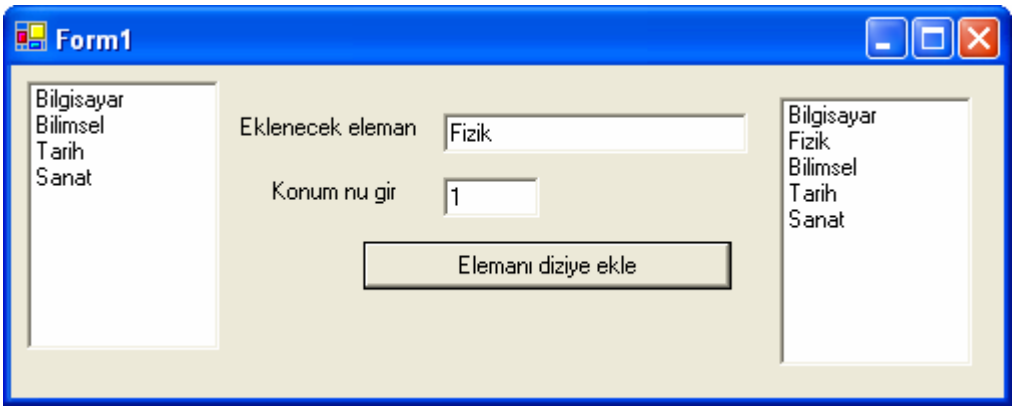
```
ArrayList kitaplar=new ArrayList();
private void Form1_Load(object sender, System.EventArgs e)
{
    kitaplar.Add("Bilgisayar");
    kitaplar.Add("Bilimsel");
    kitaplar.Add("Tarih");
    kitaplar.Add("Sanat");
    listBox1.Items.Add(kitaplar[0].ToString());
    listBox1.Items.Add(kitaplar[1].ToString());
    listBox1.Items.Add(kitaplar[2].ToString());
    listBox1.Items.Add(kitaplar[3].ToString());
}
private void button1_Click_1(object sender, System.EventArgs e)
{
    kitaplar.Insert(int.Parse(textBox2.Text),textBox1.Text);
    listBox2.Items.Add(kitaplar[0].ToString());
    listBox2.Items.Add(kitaplar[1].ToString());
    listBox2.Items.Add(kitaplar[2].ToString());
    listBox2.Items.Add(kitaplar[3].ToString());
    listBox2.Items.Add(kitaplar[4].ToString());
}
```

Program çalıştırıldığında formun tasarımı resim 2.16'daki olacaktır.



Resim 2.16: Formun tasarlanmış hali

Diziye eklenecek eleman adı ve numarası girildikten sonra dizinin son hali resim 2.17'deki gibi olur.



Resim 2.17: Insert() metoduyla diziye yeni eleman ekleme

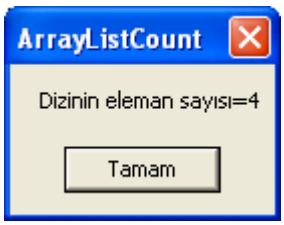
2.4.1.4. Count Özelliği

ArrayList dizisinin eleman sayısını verir.

Örnek

ArrayList sınıfıyla oluşturulan dizinin eleman sayısını bir butona tıklandığında mesaj kutusuyla ekranda gösteren program kodu aşağıda ve ekran görüntüsü resim 2.18'deki gibi olur.

```
private void button2_Click(object sender, System.EventArgs e)
{
MessageBox.Show("Dizinin eleman sayısı="
    +kitaplar.Count.ToString(),"ArrayListCount");
}
```



Resim 2.18: Dizinin eleman sayısının mesaj kutusunda gösterilmesi

2.4.1.5. TrimToSize () Özelliği

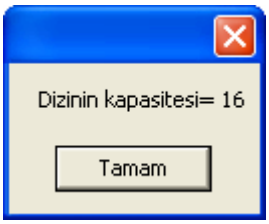
Dizi için ayrılan bellek kapasitesinin gereksiz kullanımını önlemek için boş olan alanların atılıp sadece eleman sayısı kadar alan tutmasını sağlar.

Örnek

Butona tıklandığında kitaplar dizisinin kapasitesini mesaj kutusunda gösteren program kodu aşağıdadır. Dikkat edilmesi gereken nokta, bu kodda kitaplar. TrimToSize(); satırı işletilmemiştir. Dizinin boyutunun trim edilmeden önceki hali size gösterilecektir.

```
ArrayList kitaplar=new ArrayList();
private void Form1_Load(object sender, System.EventArgs e)
{
    kitaplar.Add("Bilgisayar");
    kitaplar.Add("Bilimsel");
    kitaplar.Add("Tarih");
    kitaplar.Add("Sanat");
    listBox1.Items.Add(kitaplar[0].ToString());
    listBox1.Items.Add(kitaplar[1].ToString());
    listBox1.Items.Add(kitaplar[2].ToString());
    listBox1.Items.Add(kitaplar[3].ToString());
}
private void button3_Click(object sender, System.EventArgs e)
{
//kitaplar.TrimToSize();
MessageBox.Show("Dizinin kapasitesi= "+kitaplar.Capacity.ToString());
}
```

Dizinin boyutunun otomatik olarak verildiği 16 rakamı ekranda gösterilir (Resim 2.19).

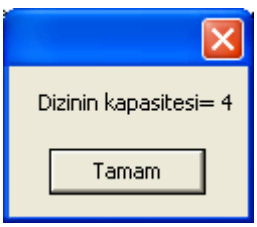


Resim 2.19: TrimToSize() metodu kullanılmadan önceki dizinin boyutu

İşletilmeyen kitaplar.TrimToSize(); satırının önündeki // karakterlerini silip projeyi tekrar çalıştırdığınızda dizi boyutunun mevcut elemanlar kadar olduğunu göreceksiniz.

```
ArrayList kitaplar=new ArrayList();
private void Form1_Load(object sender, System.EventArgs e)
{
    kitaplar.Add("Bilgisayar");
    kitaplar.Add("Bilimsel");
    kitaplar.Add("Tarih");
    kitaplar.Add("Sanat");
    listBox1.Items.Add(kitaplar[0].ToString());
    listBox1.Items.Add(kitaplar[1].ToString());
    listBox1.Items.Add(kitaplar[2].ToString());
    listBox1.Items.Add(kitaplar[3].ToString());
}
private void button3_Click(object sender, System.EventArgs e)
{
kitaplar.TrimToSize();
MessageBox.Show("Dizinin kapasitesi= "+kitaplar.Capacity.ToString());
}
```

Dolayısıyla ekran görüntüsü resim 2.20'deki gibi olacaktır.



Resim 2.20: TrimToSize() metodu kullanıldıktan sonraki dizinin boyutu

2.4.1.6. Clear () Metodu

Dizideki bütün elemanları silmek için kullanılır.

Kullanımı kitaplar.Clear(); şeklindedir. Dolayısıyla diziye aktarılan tüm elemanlar silinir.

2.4.1.7. Remove () Metodu

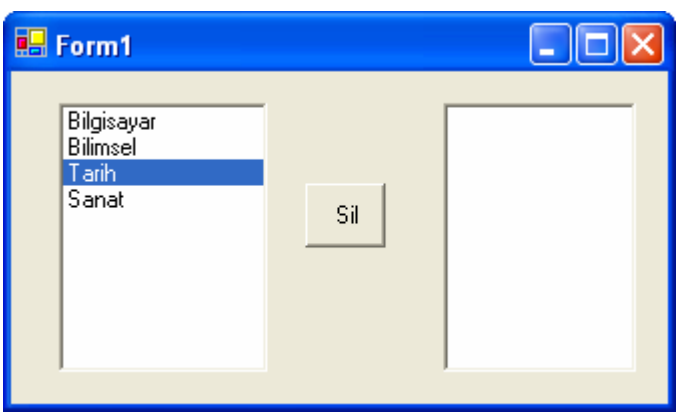
Bir elemanı diziden çıkarmak için kullanılan metottur. Numarasını verdiğiniz elemanı diziden çıkarmak için ise **RemoveAt ()** metodu kullanılır.

Örnek

ArrayList dizisiyle oluşturulan kitaplar dizisine ilk elemanları form yüklendiğinde aktarılmakta ve diziye aktarılan elemanlar birinci liste kutusunda gösterilmektedir. Liste kutusunda silmek istediğiniz bir elemanı seçip "Sil" düğmesine tıkladığınızda dizinin yeni halini ikinci liste kutusunda listeleyen program kodu şu şekilde olur.

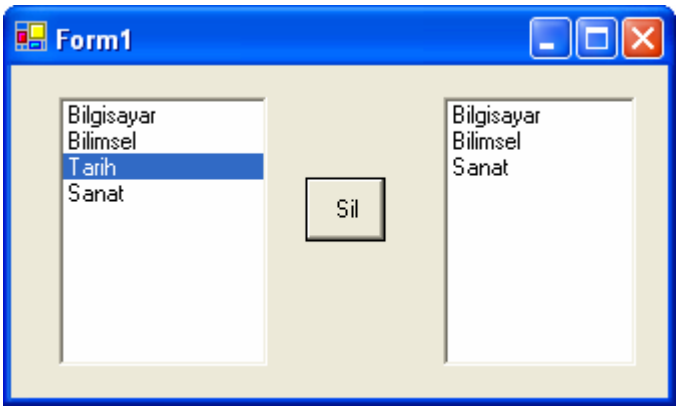
```
private void Form1_Load(object sender, System.EventArgs e)
{
    kitaplar.Add("Bilgisayar");
    kitaplar.Add("Bilimsel");
    kitaplar.Add("Tarih");
    kitaplar.Add("Sanat");
    listBox1.Items.Add(kitaplar[0].ToString());
    listBox1.Items.Add(kitaplar[1].ToString());
    listBox1.Items.Add(kitaplar[2].ToString());
    listBox1.Items.Add(kitaplar[3].ToString());
}
private void button1_Click(object sender, System.EventArgs e)
{
    kitaplar.RemoveAt((listBox1.SelectedIndex));
    for (int i=0;i<kitaplar.Count ;i++)
        listBox2.Items.Add(kitaplar[i].ToString());
}
```

Proje çalıştırıldığında formun görüntüsü resim 2.21'de olduğu gibidir.



Resim 2.21: Projenin çalıştırılmış hali

Silinecek eleman seçilip "Sil" düğmesine tıklandığında dizinin yeni hali resim 2.22' de gösterilmiştir.



Resim 2.22: Dizi elemanının RemoveAt() metoduyla silinmesi

2.4.1.8. IndexOf () Metodu

ArrayList dizisinde bir eleman aramak için kullanılan metottur. Bulma işlemi gerçekleşirse geriye elemanın index numarası döner. Arama işlemini dizinin başından itibaren değil de belli bir yerinden başlatmak isterseniz index numarasını verebilirsiniz.

dizi.IndexOf(eleman,başlangıç no);

Dizi elemanlarından belli aralıkta bulunanlarıyla arama yapmak istenirse;

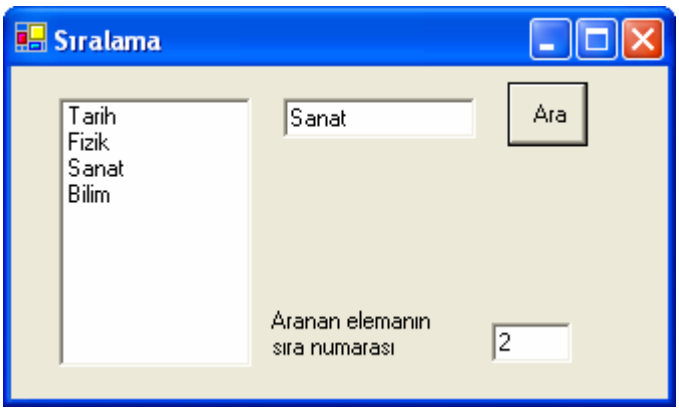
dizi.IndexOf (eleman,başlangıç no, adet); şeklinde tanımlama yapılabilir.

Örnek

Metin kutusundan girilen bilgiye göre, "Ara" düğmesine tıklandığında dizi içerisinde girilen bilgiyi arayan ve bilgiyi bulduğunda index numarasını yine bir başka metin kutusunda gösteren program kodu aşağıdaki gibidir.

```
private void Form1_Load(object sender, System.EventArgs e)
{
    kitaplar.Add("Tarih");
    kitaplar.Add("Fizik");
    kitaplar.Add("Sanat");
    kitaplar.Add("Bilim");
    listBox1.Items.Add(kitaplar[0].ToString());
    listBox1.Items.Add(kitaplar[1].ToString());
    listBox1.Items.Add(kitaplar[2].ToString());
    listBox1.Items.Add(kitaplar[3].ToString());
}
private void button1_Click(object sender, System.EventArgs e)
{
    textBox2.Text=(kitaplar.IndexOf(textBox1.Text).ToString());
}
```

Proje çalıştırıldığında elde edilen ekran görüntüsü resim 2.23'te gösterilmiştir.



Resim 2.23: Dizi içerisinde eleman arama

2.4.1.9. Sort () Metodu

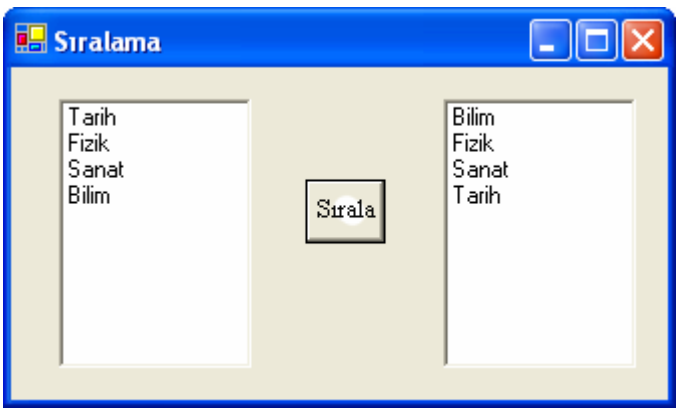
ArrayList diziyi sıralamak için kullanılan metottur.

Örnek

ArrayList kitaplar=new ArrayList(); şeklinde global olarak tanımlanan kitaplar dizisinin sıralanmamış elemanları, diziye ve liste kutusuna form yüklendiğinde aktarılmaktadır. "Sırala" düğmesine tıklandığında dizi elemanlarını alfabetik olarak sıralayıp ikinci bir liste kutusunda listeleyen program kodu aşağıda olduğu gibidir.

```
private void Form1_Load(object sender, System.EventArgs e)
{
    kitaplar.Add("Tarih");
    kitaplar.Add("Fizik");
    kitaplar.Add("Sanat");
    kitaplar.Add("Bilim");
    listBox1.Items.Add(kitaplar[0].ToString());
    listBox1.Items.Add(kitaplar[1].ToString());
    listBox1.Items.Add(kitaplar[2].ToString());
    listBox1.Items.Add(kitaplar[3].ToString());
}
private void button1_Click(object sender, System.EventArgs e)
{
    kitaplar.Sort();
    for (int i=0;i<kitaplar.Count ;i++)
        listBox2.Items.Add(kitaplar[i].ToString());
}
```

Proje çalıştırıldığında ekran görüntüsü resim 2.24'teki gibidir.



Resim 2.24: Dizinin Sort() metoduyla sıralanmış hali

2.4.1.10. Reverse () Metodu

ArrayList diziyi ters çevirmek için kullanılan metottur. Dizinin bir bölümü ters çevrilmek istenirse;

dizi.Reverse(ilk değer,adet); komut satırı yazılır.

Örnek

Arraylist dizisini ters çevirip ikinci bir liste kutusuna listeleyen program kodu şu şekilde olur.

```
private void Form1_Load(object sender, System.EventArgs e)
{
    kitaplar.Add("Tarih");
    kitaplar.Add("Fizik");
    kitaplar.Add("Sanat");
    kitaplar.Add("Bilim");
    listBox1.Items.Add(kitaplar[0].ToString());
    listBox1.Items.Add(kitaplar[1].ToString());
    listBox1.Items.Add(kitaplar[2].ToString());
    listBox1.Items.Add(kitaplar[3].ToString());
}
private void button1_Click(object sender, System.EventArgs e)
{
    kitaplar.Reverse();
    listBox2.Items.Add(kitaplar[0].ToString());
    listBox2.Items.Add(kitaplar[1].ToString());
    listBox2.Items.Add(kitaplar[2].ToString());
    listBox2.Items.Add(kitaplar[3].ToString());
}
```

Resim 2.25'teki görünüm "Ters çevir" düğmesine tıklandıktan sonra dizinin ters çevrilmiş halidir.

Resim 2.25: Dizinin ters çevrilmiş hali

2.4.2. Diğer Koleksiyonlar

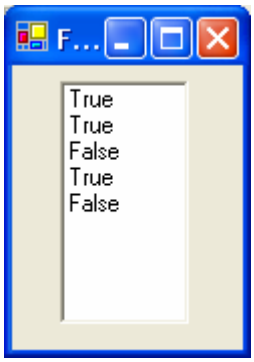
2.4.2.1. BitArray

Byte veya boolean değişken türünde değerleri içeren koleksiyondur. Örneğin mantıksal işlemlerin tümü bir yerde bulundurulmak istenirse bu koleksiyon kullanılır.

Örnek

Mantıksal bir durum dizisi ilk değerleri verilerek public olarak oluşturulmaktadır. Dizi elemanlarını liste kutunda form yüklendiğinde gösteren program kodu aşağıda ve ekran görüntüsü resim 2.26'dadır.

```
bool [] durum=new bool [](true,true,false,true,false);
private void Form1_Load(object sender, System.EventArgs e)
{
 BitArray liste=new BitArray (durum);
    for (int i=0;i<5;i++)
        listBox1.Items.Add(liste[i].ToString());
}
```



Resim 2.26: BitArray dizi tanımı ve içeriğinin yazdırılması örneği

2.4.2.2. Hashtable

Dizi elemanlarına anahtar bir değerle ulaşılmak istendiğinde kullanılan koleksiyondur. Bu koleksiyonda veriler **key/value** denilen anahtar/değer çiftleri şeklinde tutulur.

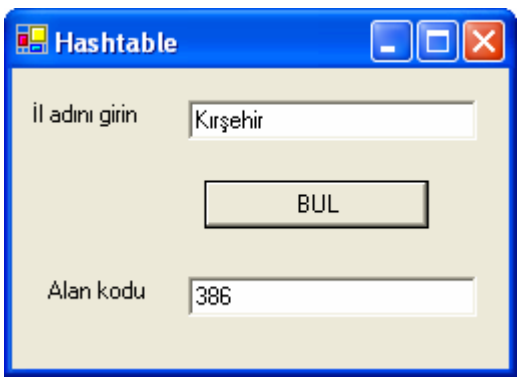
Örneğin şehir adlarını tutan bir Hashtable koleksiyonuna veriler alan kodu/şehir adı şeklinde girilebilir. Anahtar için seçilen bilgi her eleman için ayrı olmalıdır. Bu değerlere key değerleri denir. Key değerleri tektir, değiştirilemez. Key değerlerine null değerler atanamaz.

Örnek

Hashtable sınıfından oluşturulan liste dizisine form yüklendiğinde anahtar/değer elemanları eklenmiştir. "BUL" düğmesine tıklandığında, metin kutusundan girilen anahtara göre değeri bulan ve ikinci bir metin kutusunda yazdıran program kod satırları şöyledir.

```
Hashtable liste=new Hashtable();
private void Form1_Load(object sender, System.EventArgs e)
{
    liste.Add("Ankara","312");
    liste.Add("İstanbul","212");
    liste.Add("Kırşehir","386");
    liste.Add("Antalya","242");
    liste.Add("İzmir","232");
}
private void button1_Click(object sender, System.EventArgs e)
{
    object key=textBox1.Text;
    bool durum=liste.ContainsKey(key);
    if(durum==true)
    {
        textBox2.Text=liste[key].ToString();
    }
    else
        MessageBox.Show("Alan kodu bulunamadı");
}
```

Proje çalıştırılıp ilin adı girilerek BUL düğmesine tıklandığında o ilin telefon alan kodu gösterilecektir.



Resim 2.27: Hashtable sınıfının kullanımı örneği

2.4.2.3. SortedList

Hashtable koleksiyonunda olduğu gibi SortedList koleksiyonunda da key/value sistemi kullanılır. Bu koleksiyonda bilgiler sıralıdır. SortedList'in farkı, hem key hem de value değerlerine göre bilgiye erişim sağlamasıdır.

Örnek

SortedList sınıfıyla oluşturulan liste dizisinde metin kutusuna girilen il adının alan kodunu bulan ve yine bir metin kutusunda yazdıran program kodu aşağıdaki gibidir.

```
SortedList liste=new SortedList();
private void Form1_Load(object sender, System.EventArgs e)
{
    liste.Add("Ankara","312");
    liste.Add("Antalya","242");
    liste.Add("Kırşehir","386");
    liste.Add("Muğla","252");
    liste.Add("İstanbul","216");
    int toplam=liste.Count;
    for(int i=0;i<toplam;i++)
    {
        listBox1.Items.Add(liste.GetKey(i).ToString());
    }
}
private void button1_Click(object sender, System.EventArgs e)
{
    object key=textBox1.Text;
    bool durum=liste.ContainsKey(key);
    if(durum==true)
        textBox2.Text=liste[key].ToString();
}
```

Resim 2.28: SortedList sınıfının kullanımı örneği

2.4.2.4. Stack ve Queue

İçindeki bilgileri son giren ilk çıkar (Last In Fırst Out -LIFO) sistemiyle tutan koleksiyona **Stack**, ilk giren ilk çıkar (First InnFirst Out -FIFO) sistemiyle tutan koleksiyona da **Queue** adı verilir.

Bu koleksiyonları kullanmanın faydası eleman sayısı belirtilmediği taktirde koleksiyon boyutunun otomatik olarak ayarlanmasıdır. Stack koleksiyonu default olarak 10 elemanlı bir koleksiyon dizisi oluştururken Queue koleksiyonunda ise dizi boyutu 32 elemanlıdır.

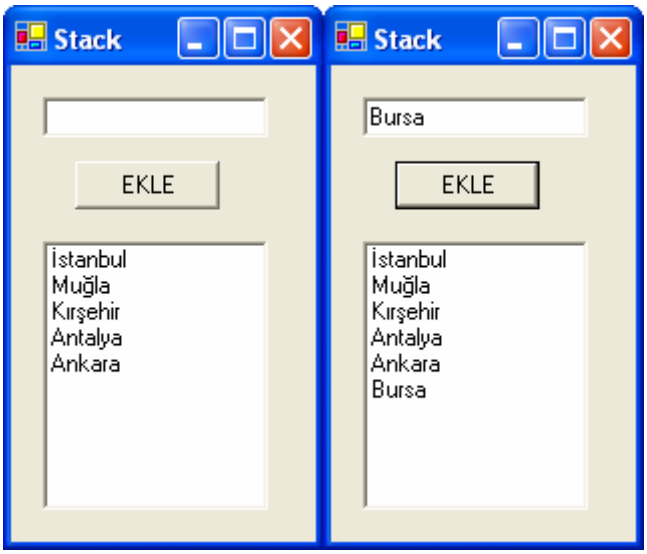
Örnek

Stack sınıfıyla oluşturulan liste dizisine "EKLE" butonuna basıldığında metin kutusundan girilen yeni elemanı diziye ekleyen program kodu aşağıdadır.

```
Stack liste=new Stack();
private void Form1_Load(object sender, System.EventArgs e)
{
    liste.Push("Ankara");
    liste.Push("Antalya");
    liste.Push("Kırşehir");
    liste.Push("Muğla");
    liste.Push("İstanbul");
    while(liste.Count>0)
    {
        listBox1.Items.Add(liste.Pop().ToString());
    }
}
private void button1_Click(object sender, System.EventArgs e)
{
    liste.Push(textBox1.Text);
    listBox1.Items.Add(liste.Pop().ToString());
}
```

Stack sınıfında diziye ekleme işlemi Push() metoduyla yapılır. Eklenen eleman dizinin üstüne yerleşir. Pop() ise, sıranın sonundaki elemanı geri döndürür ve sonra siler.

Proje çalıştırıldığında ekran görüntüsü resim 2.29'da gösterilmiştir.

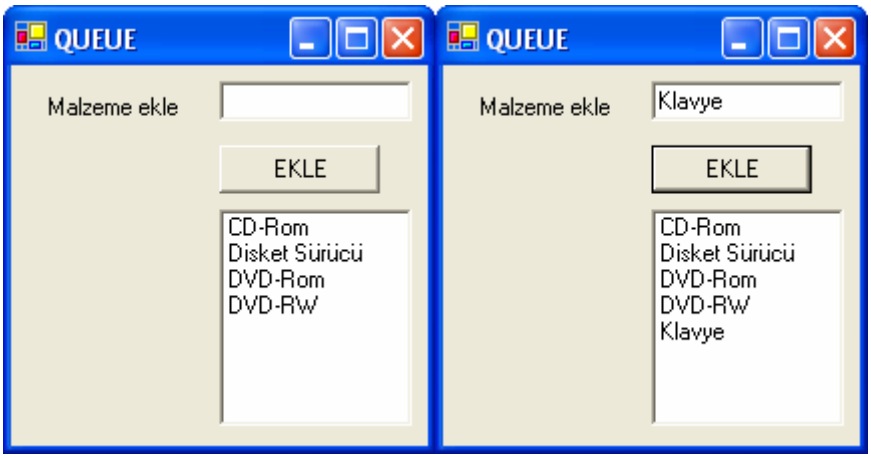


Resim 2.29: Stack sınıfının kullanımı örneği

Aynı şekilde Queue sınıfıyla oluşturulan diziye eleman eklenmesinin program kodu şöyle olur.

```
Queue liste=new Queue();
private void Form1_Load(object sender, System.EventArgs e)
{
    liste.Enqueue("CD-Rom");
    liste.Enqueue("Disket Sürücü");
    liste.Enqueue("DVD-Rom");
    liste.Enqueue("DVD-RW");
    while (liste.Count>0)
    {
        listBox1.Items.Add(liste.Dequeue().ToString());
    }
}

private void button1_Click(object sender, System.EventArgs e)
{
    liste.Enqueue ((textBox1.Text).ToString());
    listBox1.Items.Add(liste.Dequeue().ToString());
}
```

Queue sınıfında diziye eleman ekleme Enqueue() metoduyla yapılır. Eklenen eleman dizinin sonuna eklenir. Dequeue() metodu ise, sıradaki elemanı geri döndürür ve sonra siler.



Resim 2.30: Queue sınıfının kullanımı örneği

UYGULAMA FAALİYETİ-1

İşlem Basamakları	Öneriler
➢ 10 elemanlı bir sayı dizisi tanımlayınız.	➢ Dizi değişkeninin türü int olabilir. Tanımlamayı int a[10] şeklinde yapabilirsiniz.
➢ Diziye sayı girişi yapmak için bir tane döngü değişkeni, dizinin elemanlarını toplamak için başka bir tane değişken tanımlayınız.	➢ Döngü değişkeninin kapladığı alan az olduğundan döngü değişkenini byte değişken türü, dizi eleman toplamları içinde int değişken türünü verebilirsiniz.
➢ Döngü yapısı yazıldıktan sonra diziye aktarılacak sayıları bilgi giriş kutusundan (InputBox) giriniz.	➢ Bilgi giriş kutusunun parametrelerinin eksiksiz yazılmasına dikkat ediniz.
➢ Bilgi giriş kutusundan girilen sayıları işlem yapabilmek için convert ile uygun veri türüne dönüştürünüz.	➢ İşlem yapabilmek için sayılar tam sayı ya da ondalıklı değişken türlerinden birine dönüştürülmelidir.
➢ Diziye girilen sayıların toplamını hesaplatıp mesaj kutusunda yazdırınız.	➢ Hangi sayılarla hangi işlemin yapıldığını göstermek için bütün verileri gösterebilirsiniz ya da sadece sonuç değerini ekranda gösterebilirsiniz.
➢ Şimdiye kadar yazdığınız kod satırlarını gözden geçiriniz.	➢ Kod satırlarını yazarken dikkatli olunuz. Programa dillerinde her bir işaretin önemi çok büyüktür.
➢ Programı çalıştırınız. Çalışma esnasında hata oluşmuşsa kod satırlarına dönerek yazım hatalarınızı kontrol edip tekrar çalıştırınız.	➢ Amacınızı, kod satırlarını ve işlem sonucunun ekran görüntüsünü defterinize yazınız.

UYGULAMA FAALİYETİ-2

İşlem Basamakları	Öneriler
➢ Form üzerine 5 adet radyo düğmesi ekleyiniz.	➢ Radyo düğmeleri alt alta olabilir.
➢ Kod yazımını Form_Load'a yazınız.	➢ Form_Load'a kod yazmak için form üzerine fareyle çift tıklayınız.
➢ 3 elemanlı bir char dizi oluşturarak ilk değerlerini atayınız.	➢ İlk değerlerini A,B,C olarak verebilirsiniz.
➢ Char dizinin bu üç elemanını radyo düğmelerinin ilk üçünün text özelliğine atayınız.	➢ Atama esnasında ToString() metodunu kullanabilirsiniz.
➢ Char diziyi 5 elemanlı olarak program içinde yeniden tanımlayınız ve ilk değerlerini atayınız.	➢ Dört ve beşinci değer olarak D ve E değerlerini yazabilirsiniz.
➢ Dördüncü ve beşinci değeri radyo düğmelerinin 4 ve 5. sinin text özelliğine atayınız.	➢ Kod satırlarını yazarken yazım kurallarına dikkat ediniz.
➢ Programı çalıştırınız. Çalışma esnasında hata oluşmuşsa kod satırlarına dönerek yazım hatalarınızı kontrol edip tekrar çalıştırınız.	➢ Amacınızı, kod satırlarını ve işlem sonucunun ekran görüntüsünü defterinize yazınız.

UYGULAMA FAALİYETİ-3

Aşağıda verilen soruları ödev olarak yapınız. Sonuçları rapor halinde öğretmeninize sununuz.

- İlk değerleri programda tanımlanan bir int dizinin elemanlarının en büyüğünü ve en küçüğünü bulduran programın kod satırlarını ve ekran görüntüsünü yapınız.

- Komut düğmesine tıklandığında daha önce boyutu verilmeden tanımlanmış bir diziye InputBox'tan kaç sayı girişi yapılacağı sorulacaktır. Girilen sayı kadar dizi indisini yeniden tanımlayarak sayı girişi yapılıp diziye girilen sayıları liste kutusuna yazdıran program kodunu yazınız.

- Bir diziye 10 sayı girilmektedir. Girilen bu 10 sayı içinden 0' dan küçük olanları negatif dizisine, 0' dan büyük olanları pozitif dizisine aktarıp negatif ve pozitif dizilerinin elemanlarını ayrı ayrı açılır liste kutularına yazdıran program kodunu yazınız.

- 2x10 boyutlu bir dizinin birinci boyutuna sayı girişi yapılarak girilen sayıların karelerini hesaplatıp ikinci boyutuna yazdıran program kodunu yazınız.

- Bir sınıfta okuyan öğrencilerin isimleri bir diziye aktarılarak isimleri ArrayList sınıfının sort metoduna göre sıralatan programın kod satırlarını ve ekran çıktısını yapınız.

- 20 elemanlı bir sayı dizisine girilen sayıları tersine çevirerek liste kutusunda görüntüleyen programın kod satırlarını hazırlayınız.

- 10 öğrencinin okul numaraları ve boy bilgilerini diziye InputBox() metoduyla aktararak öğrencinin numarası girildiğinde boy bilgisini metin kutusuna yazdıran programı Hashtable ve SortedList sınıflarıyla ayrı ayrı yapınız.

ÖLÇME VE DEĞERLENDİRME

A. OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki soruları dikkatlice okuyarak doğru/yanlış seçenekli sorularda uygun harfleri yuvarlak içine alınız. Seçenekli sorularda ise uygun şıkkı işaretleyiniz. Boşlukları uygun şekilde doldurunuz.

1. Dizilerin eleman sayısı **new** metodu ile belirlenir.(D/Y)

2. Bir diziye farklı değişken türlerinde bilgi girişi yapılmak istenirse istenilen değişken türü kullanılabilir. (D/Y)

3. Dizinin iki boyutlu olduğunu gösteren tanımlama……şeklindedir.

4. Sütun sayısı birbirinden farklı olan dizilere…….dizi denir.

5. Bir koleksiyon çeşidi olan ArrayList koleksiyonuna belirli bir konumdan itibaren eleman eklemek istendiğinde aşağıdaki metotlardan hangisi kullanılır?
A) Add
B) IndexOf
C) Insert
D) Reverse

6. Aşağıdakilerden hangisi bir koleksiyon çeşidi değildir?
A) Hashtable
B) SortedList
C) Sort
D) Stack

DEĞERLENDİRME

Cevaplarınızı cevap anahtarı ile karşılaştırınız. Doğru cevap sayınızı belirleyerek kendinizi değerlendiriniz. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt yaşadığınız sorularla ilgili konulara geri dönerek tekrar inceleyiniz. Tüm sorulara doğru cevap verdiyseniz diğer öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ-3

AMAÇ

Nesne tabanlı programlamada birden fazla ve farklı işlemleri çeşitli özellikler ve sınırlamalar kullanarak aynı programda gerçekleştirebileceksiniz. Ayrıca hazırladığınız tek bir arayüzü istediğiniz sayıda programda kullanarak programınızın okunurluğunu artırıp fazladan kod satırları yazmak durumunda kalmayacaksınız.

ARAŞTIRMA

- Öyle bir program düşünün ki içinde birden fazla ve ayrı işlemler yapılsın. Nasıl bir programda farklı işlemler yapılabileceğini araştırınız.
- Günlük yaşantınızda kalıtım deyince aklınıza neler gelmektedir? Araştırınız.
- Sanal kelimesinin anlamını araştırınız.

3. NESNE TABANLI PROGRAMLAMANIN PRENSİPLERİ

3.1.Çok Biçimlilik (Polymorphism)

Nesne tabanlı programlamanın üç temel ögesi vardır. Bunlar, çok biçimlilik (polymorphism), kapsülleme (encapsulation) ve kalıtımdır (inheritance). Nesne tabanlı programlamada çok biçimlilik, nesnelerin dışarıdan aynı yapıda görünmelerine rağmen içerde farklı işlem yapmalarıdır. Bu nedenle sınıflar nesnelerin içindeki farklılıklardan etkilenmeden çalışır. Çok biçimlilik aslında bir arayüz (interface) kullanarak birden fazla metodun kullanılmasını sağlar. Çok biçimliliğin amacı, genel bir sınıf belirtip aynı arayüzü kullanarak oluşabilecek karmaşıklığı azaltmaktır.

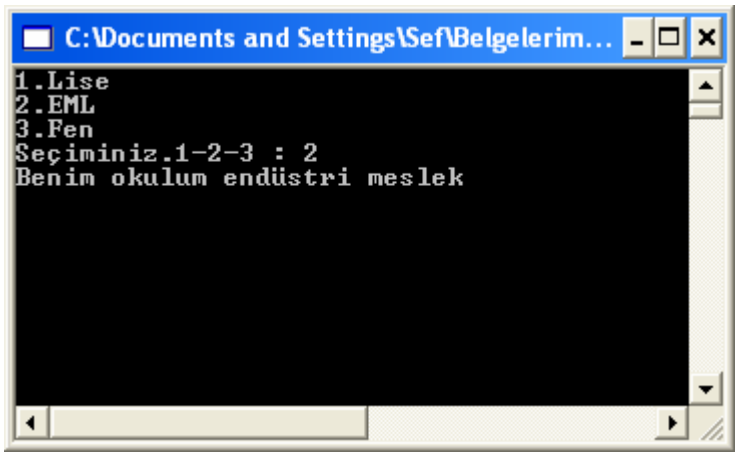
Nesne tabanlı programlamada sınıflar belli bir sıraya göre tasarlanır. Başta genel bir sınıf sonra da bu sınıftan oluşmuş ve yeni elemanlar katılarak farklı özelliklere sahip yeni sınıflar vardır. Yeni oluşturulan sınıflar temel sınıfın özelliklerini taşır. Ancak oluşturulan sınıflar temel sınıfın özelliklerini taşırlarken temel sınıf türetilen sınıfın özelliklerini taşımaz. Bir örnekle açıklayacak olursak; OKULLAR sınıfının genel bir sınıf olduğunu düşünelim. Bu sınıfın bir elemanı olan MESLEK LİSELERİ vardır. Okullar sınıfının özelliklerini meslek liselerine geçirebiliriz. Ama tam tersi söz konusu değildir. Çünkü her meslek lisesi bir okuldur ama her okul bir meslek lisesi değildir.

Örnek

Okul sınıfından Lise, Eml ve Fen sınıfları türetilerek Console'dan yapılan seçime göre okulun hangi tür olduğunu bulan programın kodları polymorphism mantığına göre yapılmıştır.

```
using System;

namespace polymorphism1
{
    class okul
    {
        class Lise:okul
        {
            public string okulum="Benim okulum düz lise";
        }
        class Eml:okul
        {
            public string okulum="Benim okulum endüstri meslek";
        }
        class Fen:okul
        {
            public string okulum="Benim okulum fen lisesi";
        }
        static void Main(string[] args)
        {
            string sec;
            Console.WriteLine("1.Lise");
            Console.WriteLine("2.EML");
            Console.WriteLine("3.Fen");
            Console.Write("Seçiminiz.1-2-3 : ");sec=Console.ReadLine();
            switch (sec)
            {
                case "1":
                        Lise L=new Lise();
                        Console.WriteLine(L.okulum);
                        Console.Read();break;
                case "2":
                        Eml ML=new Eml();
                        Console.WriteLine(ML.okulum);
                        Console.Read();break;
                case "3":
                        Fen F=new Fen();
                        Console.WriteLine(F.okulum);
                        Console.Read();break;
                        default:
                            Console.WriteLine("Sınır dışı bir seçim");break;
            }
        }
    }
}
```

Programda okul seçimi için switch() yapısı kullanılmıştır. Yapılan seçime göre (sec) program uygun satıra yönlendirilerek işlem yapılmaktadır. Herhangi bir seçim işleminde o sınıfa ait bir üye değişken oluşturularak yazma işlemi bu üye değişene (L.okulum, ML.okulum, F.okulum) göre gerçekleştirilmiştir. Buna göre programın ekran görüntüsü resim 3.1'deki gibi olur.



Resim 3.1: Polymorphism örneği

3.2. Kapsülleme (Encapsulation)

Kapsülleme, kendi amacına göre yönettiği kod ve veriyi birbirine bağlayan, kod ve veriyi dış kaynaklı karışıklık ve hatalı kullanımdan koruyan bir sistemdir. Dolayısıyla kod ve veri bir bütün olarak tutulur ve saklanır. Günlük yaşantımızda kullandığımız kapsüllü bir antibiyotiği örnek verebiliriz. Mantık aynıdır. Antibiyotiğin içindeki toz dış etkenlerden korunmak için kapsülle ambalajlanmıştır.

Nesne tabanlı programlamada kod ve veri kapsüllemeyle birbirine bağlanabilir. Yapılacak işlemler için gerekli kod ve veri kapsülün içinde vardır. Kod ve veri kapsüllemeyle birbirine bağlandığında bir nesne (object) meydana gelir.

Nesne tabanlı programlamada kapsülleme birimi sınıftır (class). Sınıf, verileri ve bu verilerle işlem yapılacak kodları belirler. Diğer bir deyişle, sınıf bir nesnenin ne şekilde yapılandırılacağını belirten planları içerir.

Kod ve veri sınıfın üyeleridir (members). Metot ise bir alt programdır. Kısaca hatırlatmak gerekirse programlama dillerinde kullanılan fonksiyonlar nesne tabanlı programlamada metot adını almıştır.

Örnek

Temel sınıfta protected erişim belirteciyle sayi adlı bir değişken oluşturularak ilk değeri atanmıştır. Temel sınıfın karakteristiğini taşıyan Tureyen sınıf oluşturularak erişim metodunda her iki sınıftan t ve tr adlı iki üye değişken tanımlanmıştır. Bu iki değişkene ilk değerleri atanarak program derlendiğinde hata meydana gelmektedir.

```
using System;
class Temel
{
    protected double sayi = 100.2;
}

class Tureyen : Temel
{
    void erisim()
    {
        Temel t=new Temel();
        Tureyen tr=new Tureyen();
        t.sayi=54.3;// Bu satırla hata olur.
        tr.sayi=55.5; //Bu satırla hata olmaz.
     }
}
```

Çünkü temel sınıftan oluşturulan t üye değişkeni türeyen sınıfta tanımlandığı için protected erişim belirteçli sayi değişkeninin içeriğini değiştiremeyecektir. Ancak, türeyen sınıftan oluşturulan tr üye değişkeni sayi değişkeninin içeriğini değiştirebilir. Böylece kapsülleşme gerçekleştirilmiş olur. Programın çalışması için t.sayi ile başlayan satırın önüne // yazarak derlemeye dahil edilmemesi gerekir.

```
using System;
namespace encaps2
{
    class Temel
    {
        protected double sayi = 100.2;
    }
    class Tureyen : Temel
    {
        static void Main()
        {
            Temel t=new Temel();
            Tureyen tr=new Tureyen();
            //  t.sayi=54.3;  // Bu satırla hata olur.
            tr.sayi=55.5; //Bu satırla hata olmaz.
        }
    }
}
```

3.3. Kalıtım (Miras alma - Inheritance)

Kalıtım nesne tabanlı programlamanın üç temel ögesinin en önemlisidir. Çünkü, kalıtım hiyerarşik sınıflandırma oluşturmayı sağlar. Konuyu biraz açmak gerekirse birbiriyle bağlantılı bir grup elemanın ortak özelliklerinden oluşan bir sınıf oluşturulabilir. Bu sınıf daha sonra yeni değişiklikler yapılarak daha özel sınıflara kalıtım yoluyla aktarılır ve her sınıf kendisinde bulunan ek özellikleri buna aktarır. Böylece uygulamanın yeni baştan yazılmasına gerek kalmaz.

Nesne tabanlı programlamada kalıtım yoluyla aktarılan sınıf, **Temel sınıf'**tır. Kalıtım işlemi yapılmış olan sınıf da oluşturulmuş (türetilmiş) sınıftır. Türetilmiş sınıf, temel sınıf tarafından tanımlanan tüm metot, özellik, operatör ve değişkenleri kalıtım yoluyla elde eder ve sadece kendisinde kullanılacak özellikleri, değişkenleri, metotları vb.yi kendisine ekler.

Kalıtımın kullanım şekli şöyledir:

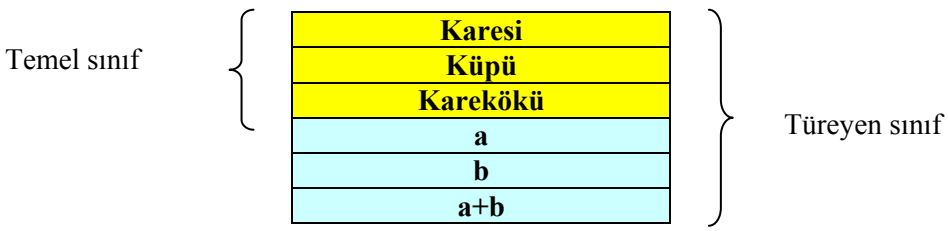
```
Namespace Kalıtım
{
  class Temel Sınıf
  {
          // Program Kodları
  }
  class Mirasçı Sınıf : Temel Sınıf
  {
          //Eklenecek metotlar
  }
}
```

Bir sınıf diğer bir sınıfı kalıtım yoluyla elde ettiğinde temel sınıfın ismi, türetilmiş sınıfın ismini takip eder ve birbirlerinden **:** (iki nokta) ile ayrılır. **:** operatörü bir sınıfın, başka bir sınıfın özelliklerini devralmasını sağlar.

Örnek

```
class Topla : Hesapla {
```

Türetilmiş Sınıf — Temel Sınıf



Tablo 3.1: Kalıtımsal gösterim (Sarı renktekiler temel sınıf, tümü ise türeyen sınıftır.)

Yeni oluşturulmuş her sınıf için sadece bir temel sınıf belirtilir. Tek bir türetilmiş sınıfa birden fazla sınıf kalıtım yoluyla aktarılamaz.

Türetilmiş olan bir sınıf başka bir türetilmiş sınıfın temel sınıfı olabilir.

Kalıtımın avantajı, temel sınıf oluşturulduktan sonra istenilen sayıda türetilmiş sınıf oluşturmaktır.

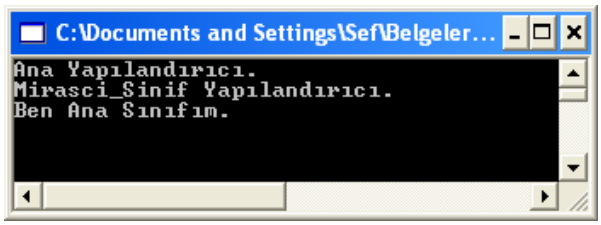
Örnek

Temel sınıfla türeyen sınıfın çalışma mantığı aşağıdaki gibidir.

```
using System;
public class Ana_Sinif
{
    public Ana_Sinif()
    {
        Console.WriteLine("Ana Yapılandırıcı.");
    }
    public void yazdir()
    {
        Console.WriteLine("Ben Ana Sınıfım.");
    }
}
public class Mirasci_Sinif : Ana_Sinif
{
    public Mirasci_Sinif()
    {
        Console.WriteLine("Mirasci_Sinif Yapılandırıcı.");
    }
    public static void Main()
    {
        Mirasci_Sinif mirasci=new Mirasci_Sinif();
        mirasci.yazdir();
        Console.ReadLine();
    }
}
```

Programda bir ana sınıf ve ana sınıftan türeyen bir mirascı sınıf olmak üzere iki sınıf oluşturulmaktadır. Program satır satır işletildiğinde, Main() metodunda mirasci adında bir üye değişken oluşturulur ve Mirasci_Sinif() metoduna yönlendirilir. Mirasçı sınıf ta ana sınıftan türetildiği için program akışı Ana_Sinif() metoduna geçer. Ana_Sinif() metodunda mesaj yazdırıldıktan sonra mirasçı sınıftaki mesaj yazdırılır. Main() metodundaki "mirasci.yazdir();" satırıyla program akışı ana sınıftaki yazdır metoduna yönlendirilerek bu metottaki mesaj yazdırılarak işlem sona erer.

Bu aşamaların sonunda çalıştırılan kodların ekran görüntüsü resim 3.2'deki gibi olacaktır.



Resim 3.2: Kalıtımın çalışma mantığının ekran görüntüsü

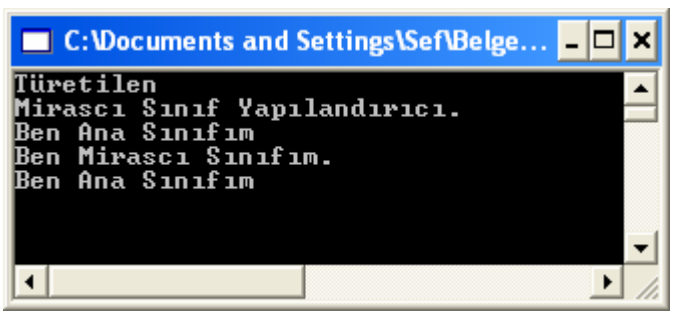
Kalıtımla ilgili bir başka örnek vermek gerekirse; aşağıdaki örneği inceleyiniz.

```
using System;
public class Ana_Sinif
{
    string anacumle;
    public Ana_Sinif()
    {
        Console.WriteLine("Ana Sınıf Yapılandırıcı.");
    }
    public Ana_Sinif(string cumlem)
    {
        anacumle = cumlem;
        Console.WriteLine(anacumle);
    }
    public void yazdir()
    {
        Console.WriteLine("Ben Ana Sınıfım");
    }
}
 public class Mirasci_Sinif : Ana_Sinif
{
    public Mirasci_Sinif() : base("Türetilen")
    {
        Console.WriteLine("Mirascı Sınıf Yapılandırıcı.");
    }
    public new void yazdir()
    {
        base.yazdir();
        Console.WriteLine("Ben Mirascı Sınıfım.");
    }
    public static void Main()
    {
        Mirasci_Sinif mirasci=new Mirasci_Sinif();
        mirasci.yazdir();
        ((Ana_Sinif)mirasci).yazdir();
        Console.ReadLine();
    }
}
```

Programda yine bir ana sınıf ve ana sınıftan türeyen bir mirasçı sınıf olmak üzere iki sınıf oluşturulmaktadır. Main() metodunda "mirasci" adında türeyen sınıfa ait bir üye değişken oluşturulmakta ve program akışı Mirasci_Sinif() metoduna yönlendirilmektedir. Mirasci_Sinif() metodunda kullanılan base komutuyla string bir ifade mirasçı sınıf işletilmeden ana sınıfa gönderilmekte ve ana sınıfta bir değişkenle bu ifade karşılanmaktadır. Karşılanan bu ifade Ana_Sinif() metodunda yazdırıldıktan sonra program akışı Mirasci_Sinif() metoduna dönmekte ve bu sınıftaki mesajın yazdırılması gerçekleşmektedir. Yazdır metodu her iki sınıfta kullanılmaktadır. Mirasçı sınıftaki yazdır metodunun new ile tanımlandığına dikkat ediniz. Böylece her iki yazdır metodu birbirinden ayrılmaktadır. "mirasci.yazdir()" satırıyla program akışı new ile tanımlanan yazdır metoduna yönlendirilir. Bu metodun ilk satırında bulunan "base.yazdir();" komut satırıyla da ana sınıf içinde bulunan yazdır metodunu çalıştırır. Ana sınıftaki yazdır metodunda bulunan mesaj yazdırıldıktan sonra program akışı "base.yazdir();" satırından sonraki satıra geçer ve new ile yeniden tanımlanan yazdır metodundaki mesajı yazar. "((Ana_Sinif)mirasci).yazdir();" satırıyla oluşturulan "mirasci" üye değişkeni kullanılarak ana sınıfın yazdır metoduna program yönlendirilir ve bu metottaki mesaj yazdırılmış olur.

Verilen bu örnekle türetilmiş bir sınıfın ana sınıfla iletişimi gösterilmektedir.

Kod satırları derlenip çalıştırıldığında ekran görüntüsü resim 3.3'te görüldüğü gibi olur.



Resim 3.3: Türetilmiş sınıf ile ana sınıfın iletişimi örneği

3.3.1. Kalıtımda Üye Erişimi

Sınıf üyelerine erişimi kısıtlamak nesne tabanlı programlamanın bir ögesidir. Çünkü, erişimin kısıtlanması nesnenin yanlış kullanımını önler. Tanımlanmış metotlarla sadece belirli verilere erişim sağlanır bu verilere uygun olmayan değerlerin atanması engellenebilir. Ayrıca, iyi bir erişim yapılarak nesnenin içindeki verilerin nasıl ve ne zaman kullanıldığını kontrol etmek mümkündür.

Üye erişimini belirli sınırlar dâhilinde yapabilmek için erişim belirteçleri kullanılır. Böylece bir metot veya değişkene belirlenen sınırlar dâhilinde ulaşılabilir. Kullanılan belirteçler şunlardır.

3.3.1.1. Public

Programın tümünden erişilmek istenen veriler public anahtar kelimesiyle birlikte kullanılır. Sadece aynı uygulamadan değil başka uygulamalardan da public üyelere erişilebilir. Public tanımlama yapılırken aynı namespace içinde olmak zorunlu değildir.

3.3.1.2. Private

Sınıf üyelerinin gereksiz yere kullanılmalarını önlemek için bu üyeler private tanımlanır. Kısaca, veriler sadece tanımlandığı sınıfta geçerli olur. Böylece türetilmiş sınıf, kendi temel sınıfının tüm üyelerini kullansa bile temel sınıftaki private üyeleri kullanamaz.

3.3.1.3. Internal

Internal erişim belirteci aynı public erişim belirteci gibidir. Tek farkı, sadece aynı uygulama içinden erişiliyor olmasıdır. Aynı uygulama içinde tüm sınıflar tarafından kullanılacak metotlar, internal olarak tanımlanabilir.

Örnek

Form üzerinde bulunan metin kutusundan girilen yaş bilgisini mesaj kutusunda gösteren program kodları aşağıdaki gibidir.

Bu uygulama bir Windows application olduğunda için sınıf kullanmak için projeye bir "Class" dâhil etmeniz gerekir.

Oluşturulan Class'a aşağıdaki kod satırlarını yazınız.

```
using System;
using System.Windows.Forms;

namespace inherit5
{
    public class Class1
    {
        internal string yyas="";
        public Class1(string veri)
        {
            this.yyas=veri;
        }
        internal void yasgoster()
        {
            MessageBox.Show("Yaşınız "+this.yyas);
        }
    }
}
```

Form'a da bir buton ekleyerek üzerinde çift tıklatınız ve button1_Click için aşağıdaki kod satırlarını yazınız.

```
private void button1_Click(object sender, System.EventArgs e)
{
    string bilgi=textBox1.Text ;
    Class1 yas=new Class1(bilgi);
    yas.yasgoster();
}
```

Butona tıklandığında metin kutusundan girilen yaş bilgisi "bilgi" değişkenine aktarılacak ve "yas" üye değişkeniyle oluşturduğumuz Class' a gönderilecektir. Class'ta tanımlanan "yyas" değişkeni internal tipte olup aynı uygulama içerisinde kullanılabilecektir. "yas" üye değişkeniyle gelen bilgi Class'ta "veri" ile karşılanarak "yyas" internal değişkenine aktarılacak ve Class'ta bulunan yasgoster() metoduyla da girilen yaş bilgisi mesaj kutusuyla ekranda gösterilecektir.

Resim 3.4: Internal erişim kullanımı örneği

3.3.1.4. Protected

Temel sınıfın private üyesine türetilmiş sınıfın erişemediği bilinmektedir. Ancak protected (korumalı) bir üye oluşturularak bu özellik değiştirilebilir. Sınıfın bir üyesi protected olursa bu üye aslında bir istisna dışında private'tir. Protected üye kalıtımla aktarıldığında bu istisna oluşur. Böylece temel sınıfın protected üyesi türetilen sınıfın protected üyesi olur ve türetilen sınıftan erişilir.

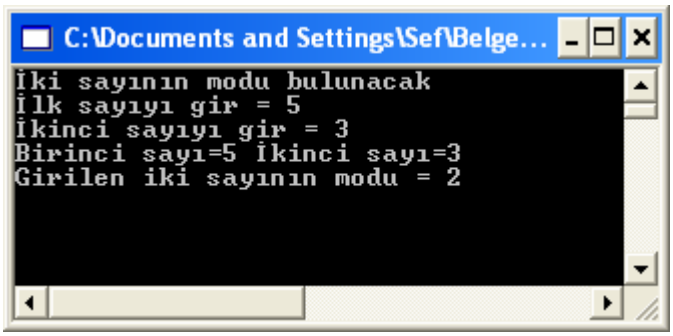
Böylece, kendi sınıfında private olan, kalıtım yoluyla aktarılabilen ve türetilmiş sınıflardan erişebilen üyeler oluşturulabilir.

Örnek

Konsoldan girilen iki sayının "modu" bulunarak yine konsola yazdıran program kod satırları aşağıdadır.

```
using System;
class Temel
{
  protected int x,y;
  public void ayarla(int p, int q)
  {
    x=p;
    y=q;
  }
  public void goster()
  {
    Console.WriteLine("Birinci sayı="+x+ " "+"İkinci sayı=" +y);
  }
}
class Tureyen:Temel
{
  int s;
  public void ayarla1()
  {
    s=x%y;
  }
  public void goster1()
  {
    Console.WriteLine("Girilen iki sayının modu = "+s);
  }
}
class Ana
{
  public static void Main()
  {
    Tureyen eleman=new Tureyen();
    Console.WriteLine("İki sayının modu bulunacak");
    Console.Write("İlk sayıyı gir = ");
    int sayi1;
    sayi1=int.Parse(Console.ReadLine());
    Console.Write("İkinci sayıyı gir = ");
    int sayi2;
    sayi2=int.Parse(Console.ReadLine());
    eleman.ayarla(sayi1,sayi2);
    eleman.goster();
    eleman.ayarla1();
    eleman.goster1();
    Console.ReadLine();
  }
}
```

Program, önce temel sınıftaki metotları, ardından türetilmiş sınıftaki metotları çalıştıracaktır.



Resim 3.5: Protected erişim belirleyicisi örneği

3.3.2. Base Kullanımı

Base komutu, türetilmiş sınıflarda temel sınıftan bir metodu veya değişkeni çağırmak amacıyla kullanılır.

Örnek

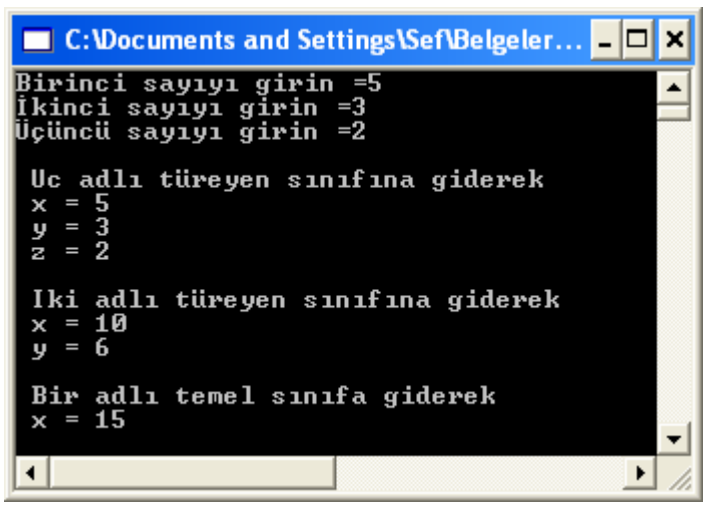
Konsoldan girilen int tipindeki üç sayının;

- İçeriklerini yazan,
- Bir ve ikinci sayının iki katını yazan,
- Birinci sayının üç katını yazan program kodu örneği aşağıdadır.

```
using System;
class Bir
{
  public int x;
  public Bir(int x)
  {
    this.x = x;
  }
}
class Iki :Bir
{
  public int y;
  public Iki(int x, int y):base(x)
  {
    this.y = y;
  }
}
class Uc :Iki
{
  public int z;
  public Uc(int x, int y,int z):base(x,y)
  {
    this.z = z;
  }
}
```

```
class Base1
{
  public static void Main()
  {
    Console.Write("Birinci sayıyı girin =");
    int sayi1=Convert.ToInt32(Console.ReadLine());
    Console.Write("İkinci sayıyı girin =");
    int sayi2=Convert.ToInt32(Console.ReadLine());
    Console.Write("Üçüncü sayıyı girin =");
    int sayi3=Convert.ToInt32(Console.ReadLine());

    Uc gonder=new Uc(sayi1,sayi2,sayi3);
    Console.WriteLine();
    Console.WriteLine(" Uc adlı türeyen sınıfına giderek  ");
    Console.WriteLine(" x = " + gonder.x);
    Console.WriteLine(" y = " + gonder.y);
    Console.WriteLine(" z = " + gonder.z);
    Console.WriteLine();

    Iki gonder1 = new Iki(sayi1*2,sayi2*2);
    Console.WriteLine(" Iki adlı türeyen sınıfına giderek ");
    Console.WriteLine(" x = " + gonder1.x);
    Console.WriteLine(" y = " + gonder1.y);
    Console.WriteLine();

    Bir gonder2 = new Bir(sayi1*3);
    Console.WriteLine(" Bir adlı temel sınıfa giderek ");
    Console.WriteLine(" x = " + gonder2.x);
    Console.WriteLine();
    Console.ReadLine();
  }
}
```

Programın çalışma mantığı en son türetilen Uc" sınıfına gelen üç sayının sonuncusu alınıp diğer ikisi bir üst sınıfa; "Ikı" sınıfına gelen iki sayının sonuncusu alınıp diğeri "Bir" sınıfına gönderilmektedir.



Resim 3.6: Birbirinden türetilmiş sınıflarda Base komutu kullanımı

3.3.3. Sanal Metot Tanımlamak (Virtual Metot)

Sanal metot kalıtım alan sınıflarda temel sınıfa ait bir metodu aynı isimle tanımlamak için kullanılır. Aslında oluşturulan her sanal metot, türetilmiş sınıfın bir versiyonudur. Nesne tabanlı programlama derleyicisi çalışma zamanında hangi versiyonun çağrılacağını belirler. Sanal metodun hangi versiyonunun çalıştırılacağını belirleyen, referansta bulunan nesnenin tipidir. Böylece farklı nesne tipleriyle sanal metodun farklı versiyonları çalıştırılır.

Bir temel sınıf içinde bir metot sanal olarak tanımlanmak istenirse metodun önüne **virtual** kelimesi yazılmalıdır. Sanal metot aynen normal metot gibi çağrılır. Ayrıca, sanal metotlar static veya abstract olarak belirtilmez.

3.3.4. Override Metot Tanımlamak

Temel sınıftan türetilmiş bir sınıfta, temel sınıfa ait bir metodu aynı adla tanımlamak için kullanılır. Sanal metot türetilmiş sınıf tarafından yeniden tanımlanırken **override** niteleyicisi kullanılır. Override niteleyicisi sanal (virtual) ve özet (abtract) metotlarla kullanılmaktadır.

Sanal metot türetilmiş sınıfta yeniden tanımlanırsa, buna metodu devre dışı bırakma (üzerine bindirme) adı verilir.

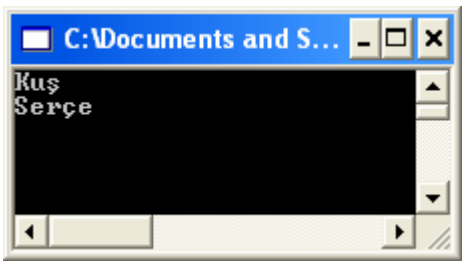
Örnek

Bu örnekte "Ucan_Kus" adlı bir ana sınıf ve bu ana sınıftan türemiş "Yabani" ve "Evcil" sınıfları vardır. Main () metodunda "Evcil" türemiş sınıfından bir üye değişken (mavis) oluşturularak Cins() metoduna gönderilmektedir. Türemiş sınıf olan "Evcil" sınıfından da base komutuyla ana sınıftaki metoda ulaşılmıştır.

```
using System;
class Ucan_Kus
{
    public virtual  void Cins()
    {
        Console.WriteLine("Kuş");
    }
}
class Yabani:Ucan_Kus
{
    public override void Cins()
    {
        Console.WriteLine("Kartal");
    }
}
```

```
class Evcil:Ucan_Kus
{
     public override void Cins()
    {
        base.Cins();
        Console.WriteLine("Serçe");
    }
}
class Ana_Prog
{
    static void Main()
    {
        Evcil mavis=new Evcil();
        mavis.Cins();
        Console.ReadLine();
    }
}
```

Sanal (virtual) metot tanımlanarak bu metodun türetilmiş sınıflarda tekrar tanımlanabileceği gösterilmiştir. Türetilmiş sınıflarda aynı metot tanımlanırken "override" ifadesi kullanılarak türetilmiş sınıftaki metodun temel sınıfta var olduğu ancak, türetilmiş sınıftaki metodun temel sınıftaki metodu etkisiz yaptığı görülmüştür.



Resim 3.7: Sanal (virtual) ve üzerine bindirme (override) metoduyla ilgili ekran görüntüsü

3.3.5. Kalıtım Vermeyi Engellemek (Sealed)

Bir sınıfın başka sınıflara kalıtım vermesini engellemek için sealed komutu kullanılır. Bu komut kullanıldığında sınıfınızdan başka hiçbir sınıfa kalıtım alınamaz. Ancak, yeni türetilen bir değişken (üye değişken) yardımıyla sınıfın metotları çağrılabilir.

Sealed komutunun kullanım şekli şöyledir:

```
sealed class Temel
{
   // Program kod satırları
}

//
class Türeyen : Temel
{
   // HATALIDIR!!! Temel sınıftan bir sınıf türetilemez.

   // Program kod satırları
}
```

Temel sınıf, kalıtım yoluyla sealed komutundan dolayı türeyen sınıfına aktarılamaz.

3.3.6. Özet Sınıfların Kullanımı (Abstract)

Özet sınıf, **abstract** (özet) tip niteleyicisi verilerek oluşturulur. Özet sınıfın detayları yoktur. Özet sınıfın detayları yani yapılacak işlemleri bulunmadığı için türetilmiş sınıflarda devre dışı kalır. Özet sınıflar otomatik olarak sanaldır . "virtual" ve "abstract" terimleri birlikte kullanılmaz.

Abtract özelliği yalnızca normal metotlarda kullanılır. Static metotlarla kullanılmaz.

Bir sınıfta bir veya birden fazla özet metot kullanılıyorsa bu sınıf abstract belirleyicisiyle tanımlanmalıdır.

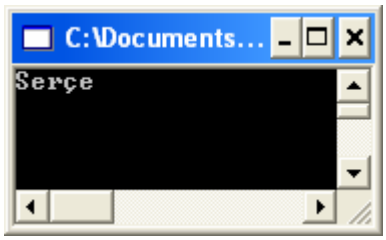
"new" metodu kullanılarak özet sınıfa ait bir nesne oluşturulamaz.

Bir özet sınıf, kalıtım yoluyla türetilmiş sınıfa aktarılabilir. Bu durumda, türetilen sınıf, temel sınıf içindeki tüm özet metotları uygulamak zorundadır. Ayrıca türetilen sınıftan nesne oluşturulabilir.

Örnek

Programda "abstract" olarak tanımlı bir ana sınıf ve bir türemiş sınıf kullanılmıştır. Sınıfın sadece türetilebileceğini belirtmek için ana sınıfın başına abstract yazılmıştır. Türetilen sınıflarda da metodun içinin doldurulabilmesi için ana sınıftaki Cins() metodu abstract olarak tanımlanmıştır.

```
using System;
abstract class Ucan_Kus
{
    public abstract void Cins();
}
class Evcil:Ucan_Kus
{
    public override void Cins()
    {
        Console.WriteLine("Serçe");
    }
}
class Ana_Prog
{
    static void Main()
    {
        Evcil mavis=new Evcil();
        mavis.Cins();
        Console.ReadLine();
    }
}
```



Resim 3.8: "abstract" örneği ekran görüntüsü

3.3.7. Object Sınıf

Özel bir sınıftır. Tüm sınıflar ve diğer tipler için kullanılabilen yapıdadır. Object tipteki referans değişkeni herhangi bir tipteki nesneye referans olabilir.

3.4. Arayüz (Interface)

Arayüz, bir sınıf tarafından uygulanacak olan birden fazla metodu tanımlamak için kullanılır. Arayüz, herhangi bir metot uygulamadan uygulamanın nasıl yapılacağını belirten mantıksal bir özelliktir. Kısaca, kullanıcıyla uygulama arasındaki iletişimi en kolay ve en anlaşılır yapan bir özelliktir.

Arayüz yapısal olarak özet sınıflara benzer. Arayüz bir kez tanımlandıktan sonra istenilen sayıda sınıf bu arayüzü kullanabilir. Ayrıca, bir sınıf da istenilen sayıda arayüzü uygulayabilir.

Arayüz tanımlamak için interface kelimesi kullanılır.

Arayüz tanımlama yapısı şöyledir:

```
interface Ad
{
  dönüş_tipi metot_adı1(değişkenler);
  dönüş_tipi metot_adı2(değişkenler);
  //
  dönüş_tipi metot_adıN(değişkenler);
}
```

Arayüzler dönüş tipleri, isim ve kullanılan değişkenlerle tanımlanır. Arayüz içindeki metotlarda uygulama yoktur. Bu nedenle, interface içeren sınıflar metotların tümünü uygular. Arayüz içindeki metotlar her yerden ulaşılabileceği için **public'**tir. Bu yüzden açıkça bir erişim belirleyicisi kullanılmasına izin verilmez. Ayrıca, arayüz üyeleri static olarak tanımlanmaz.

Arayüzlerin programda kullanım şekli şöyledir:

```
class sınıf_adı: arayüz_adı
  {
        // sınıf kodları
  }
```

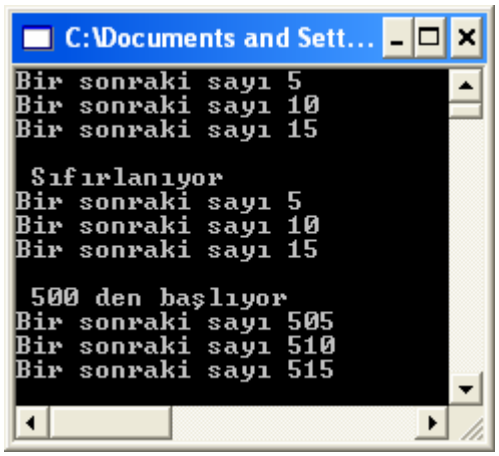
Bu kullanımda sınıf arayüzün tümünü uygular. Birden fazla arayüz kullanılacaksa arayüz adları arasına, (virgül) işareti konur. Arayüzler tanımlanırken isimlerinin başına "I" harfi konur. Bunun sebebi, arayüzlerle türetilmiş sınıfları birbirinden ayırmaktır.

Örnek

Bu program, başlangıç ve artış değeri verilen sayıları üç kere ekrana yazdıran bir programdır. Programda mevcut Class'ın dışında bir arayüz oluşturulmuştur. Class'tan "sayi" üye değişkeni oluşturularak yine Class'la aynı adı taşıyan metoda gönderilmekte ve başlangıç değerleri atanarak ekrana yazdırılmaktadır. Arayüzün nasıl tanımlandığına dikkat ediniz.

```
using System;
namespace interface3
{
  public interface Isayilar
  {
    int Sonraki();
    void Sifirla();
    void Baslangic_ayarla(int a);
  }
  class Class1:Isayilar
  {
    int basla;
    int deger;
    public Class1()
    {
        basla=0;
        deger=0;
    }
    public int Sonraki()
    {
        deger +=5;
        return deger;
    }
    public void Sifirla()
    {
        deger=basla;
    }
    public void Baslangic_ayarla(int a)
    {
        basla=a;
        deger=basla;
    }
    public static void Main(string[] args)
    {
        Class1 sayi=new Class1();
        for(int x=0;x<3;x++)
                Console.WriteLine("Bir sonraki sayı "+ sayi.Sonraki());
        Console.WriteLine("\n Sıfırlanıyor");
        sayi.Sifirla();
        for(int x=0;x<3;x++)
                Console.WriteLine("Bir sonraki sayı "+ sayi.Sonraki());
        Console.WriteLine("\n 500 den başlıyor");
        sayi.Baslangic_ayarla(500);
        for(int x=0;x<3;x++)
                Console.WriteLine("Bir sonraki sayı "+ sayi.Sonraki());
        Console.Read();
    }
  }
}
```

Kod satırları derlenip çalıştırıldığında ekran görüntüsü resim 3.9'daki gibidir.



Resim 3.9: Interface kullanılarak yapılan örneğin ekran görüntüsü

3.4.1. Arayüz (Interface) Özellikleri

Arayüzde kullanılacak özellikler aynı metotlarda olduğu gibi arayüzün içinde kod satırları olmadan belirtilebilir.

Özelliklerin kullanımı şu şekildedir:

```
Tip isim
  {
     get;
     set;
  }
```

"get" ve "set" deyimleri bir değişkenin değerini almak ve ayarlamak için kullanılır. "set" deyimiyle ilgili elemana değer aktarılmakta ve "get" bloğunda ise aktarılmış olan değer elemana gönderilmektedir.

Örnek

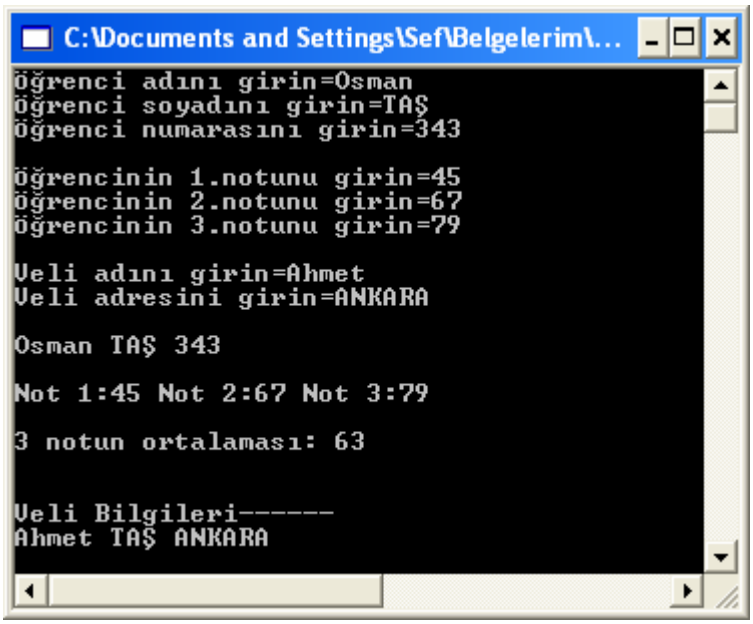
Bu programda bir öğrencinin kişisel, veli ve ders notu bilgileri üç ayrı arayüzle tanımlanarak konsoldan girilmektedir. Girilen bilgileri ve ders notu ortalamasını ekrana yazdıran program kodu aşağıdaki gibidir. Programdaki arayüzlerle diğer kod satırları yazılmadan bir şablon oluşturulmuş ve programın okunurluğu artırılarak programın akıcılığı sağlanmıştır.

```
using System;
namespace Interfaces4
{
    public interface IOgrenci
    {
        void OgrenciBilgi();
        string OgrAd{get;set;}
        string OgrSoyad{get;set;}
        string OkulNo{get;set;}
    }
    public interface IVeli
    {
        string VeliAd{get;set;}
        string VeliSoyad{get;set;}
        string VeliAdres{get;set;}
        void VeliBilgi();
    }
    public interface IOgrnot
    {
        int Ogrnt1{get;set;}
        int Ogrnt2{get;set;}
        int Ogrnt3{get;set;}
        void OgrenciNot();
    }
   public class Ogrenci:IOgrenci,IOgrnot,IVeli
   {
       private int nt1,nt2,nt3;
       private string OAd,OSoyad,ONo,VAd,VSoyad,VAdres;
       public string OgrAd
       {
           get{return OAd;}
           set{OAd=value;}
       }
       public string OgrSoyad
       {
           get{return OSoyad;}
           set{OSoyad=value;}
       }
       public string OkulNo
       {
           get{return ONo;}
           set{ONo=value;}
       }
       public void OgrenciBilgi()
       {
           Console.WriteLine();
           Console.WriteLine(OAd+" "+OSoyad+" "+ONo);
       }
```

```
        public string VeliAd
        {
            get{return VAd;}
            set{VAd=value;}
        }
        public string VeliSoyad
        {
            get{return VSoyad;}
            set{VSoyad=value;}
        }
        public string VeliAdres
        {
            get{return VAdres;}
            set{VAdres=value;}
        }
        public void VeliBilgi()
        {
            Console.WriteLine();
            Console.WriteLine("Veli Bilgileri------");
            Console.WriteLine(VAd+" "+VSoyad+" "+VAdres);
        }
    public int Ogrnt1
    {
        get{return nt1;}
        set{nt1=value;}
    }
    public int Ogrnt2
    {
        get{return nt2;}
        set{nt2=value;}
    }
    public int Ogrnt3
    {
        get{return nt3;}
        set{nt3=value;}
    }
    public void OgrenciNot()
    {
        Console.WriteLine();
        Console.WriteLine("Not 1:"+nt1+" Not 2:"+nt2+" Not 3:"+nt3);
        Console.WriteLine();
        Console.WriteLine("3 notun ortalaması: "+((nt1+nt2+nt3)/3));
        Console.WriteLine();
    }
}
```

```
    class Class1
    {
        static void Main(string[] args)
        {
            Ogrenci Ogr1=new Ogrenci();
            Console.Write("Öğrenci adını girin=");
            Ogr1.OgrAd =Console.ReadLine();
            Console.Write("Öğrenci soyadını girin=");
            Ogr1.OgrSoyad=Console.ReadLine();
            Console.Write("Öğrenci numarasını girin=");
            Ogr1.OkulNo=Console.ReadLine();
            Console.WriteLine();
            Console.Write("Öğrencinin 1.notunu girin=");
            Ogr1.Ogrnt1=int.Parse(Console.ReadLine());
            Console.Write("Öğrencinin 2.notunu girin=");
            Ogr1.Ogrnt2=int.Parse(Console.ReadLine());
            Console.Write("Öğrencinin 3.notunu girin=");
            Ogr1.Ogrnt3=int.Parse(Console.ReadLine());
            Console.WriteLine();
            Console.Write("Veli adını girin=");
            Ogr1.VeliAd=Console.ReadLine();
            Ogr1.VeliSoyad=Ogr1.OgrSoyad;
            Console.Write("Veli adresini girin=");
            Ogr1.VeliAdres=Console.ReadLine();
            Ogr1.OgrenciBilgi();
            Ogr1.OgrenciNot();
            Ogr1.VeliBilgi();
            Console.Read();
        }
    }
}
```

Programda önce öğrencinin ad, soyad ve okul numarası bilgileri istenmektedir. Alınan bu bilgiler üç arayüzden oluşturulan "Ogrenci" sınıfındaki ilgili metotlara yönlendirilmektedir. Yapılan işlemlerden sonra "OgrenciBilgi()", "OgrenciNot()" ve "VeliBilgi()" metotları kullanılarak elde edilen bilgiler ekrana yazdırılmaktadır.



Resim 3.10: Arayüzde get-set kullanılarak yapılan örneğin ekran görünümü

UYGULAMA FAALİYETİ-1

Seçilen mevsime göre o mevsimde hangi faaliyetin yapıldığını bulan programı yapınız.

İşlem Basamakları	Öneriler
➢ Mevsimler için bir sınıf tanımlayınız.	➢ Sınıf adı "Mevsimler" olabilir.
➢ Mevsim adlarını "Mevsimler" sınıfından türeterek tanıtınız.	➢ "Class Ilkbahar : Mevsimler" örneğinde olduğu gibi diğer mevsimleri de türetebilirsiniz.
➢ Bu sınıflar içerisinde bir değişken kullanarak o mevsimde yapılacak faaliyeti yazınız.	➢ Değişkeni tüm programda kullanmak için public tanımlayınız.
➢ Main() metodunda mevsim adlarını ekrana yazdırınız ve bir mevsim seçilmesini sağlayınız.	➢ Yapılan seçimi "sec" isimli değişkene aktarabilirsiniz.
➢ Yapılan seçime göre program akışını yönlendirecek bir yapı kullanınız.	➢ Switch() yapısını kullanabilirsiniz.
➢ Her seçim için sınıflarına uygun üye değişkenler tanımlayınız.	➢ Üye değişkenler için sınıf isimlerinin kısaltmalarını kullanarak programın anlaşılırlığını artırabilirsiniz.
➢ Üye değişkeni kullanarak ait olduğu sınıftaki faaliyeti yazdırınız.	➢ Console.WriteLine (Ilk.faaliyet); örneğinde olduğu gibi.
➢ Programı çalıştırınız. Çalışma esnasında hata oluşmuşsa kod satırlarına dönerek yazım hatalarınızı kontrol edip tekrar çalıştırınız.	➢ Amacınızı, kod satırlarını ve işlem sonucunun ekran görüntüsünü defterinize yazınız.

UYGULAMA FAALİYETİ-2

Aşağıda verilen soruları ödev olarak yapınız. Sonuçları rapor halinde öğretmeninize sununuz.

- Console'dan girilen taban ve yükseklik değerlerine göre üçgenin alanını bulan programın kod satırlarını yazınız.

- Bir işçinin kişisel ve çalışma saatleriyle ilgili bilgilerini ayrı ayrı arayüzlerde tanımlayınız. Console'dan işçinin ad, soyad, adres bilgileri, çalıştığı toplam saat ve saat ücreti bilgileri girilerek işçinin bir ayda alacağı maaşını hesaplatan ve yazdıran programı yapınız.

ÖLÇME VE DEĞERLENDİRME

A. OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki soruları dikkatlice okuyarak doğru/yanlış seçenekli sorularda uygun harfleri yuvarlak içine alınız. Seçenekli sorularda ise uygun şıkkı işaretleyiniz. Boşlukları uygun şekilde doldurunuz.

1. Çok biçimlilik birden fazla işlemi ayrı ayrı metotlarla yaptırma işidir.(D/Y)

2. Aşağıdakilerden hangisi erişim belirteci değildir?
A) base
B) public
C) protected
D) internal

3. Verilerin sadece tanımlı olduğu sınıfta geçerli olması için kullanılan erişim belirteci hangisidir?
A) private
B) public
C) protected
D) internal

4. Bir temel sınıftan sadece bir sınıf türetilebilir. (D/Y)

5. new metoduyla özet sınıfta bir nesne oluşturulamaz. (D/Y)

6. Arayüz içinde kullanılan metotlar……..'tir.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarı ile karşılaştırınız. Doğru cevap sayınızı belirleyerek kendinizi değerlendiriniz. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt yaşadığınız sorularla ilgili konulara geri dönerek tekrar inceleyiniz. Tüm sorulara doğru cevap verdiyseniz diğer öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ–4

AMAÇ

Metodlara fonksiyonellik kazandırmak için temsilci ekleyip programda kullanabileceksiniz.

ARAŞTIRMA

- Günlük hayatta temsilci nedir? Araştırınız.
- Temsilcilerin görevleri nedir? Araştırınız.

4. TEMSİLCİ

Metotların birbirine referansta bulunması için kullanılan nesnelere temsilci (delege-delegate) denir. Başka bir deyişle, bir veya birden fazla metodu program içinde temsil eden referans türü nesnelerdir.

Temsilciler sınıflar içinde oluşturulmuş olan prosedür ve fonksiyonları, aynı veya başka sınıf içerisinde çağırabilmek ve fonksiyon veya prosedürlerin içindeki işlemleri gerçekleştirmek amacıyla oluşturulur.

Metotların bellekte bir adresi vardır. Metotların bellekteki adresleri tutulmak istendiğinde temsilciler kullanılabilir. Bir temsilciyle metodun adresi tutulduğunda artık metodun çalıştırılması temsilcinin çağrılmasıyla da gerçekleştirilebilir.

Bir temsilci ayrı ayrı metotları çağırmak için kullanılabilir.

Bir temsilciyi, bir metodu göstermesi için kullanmak gerekirse çalışma anında onu **new** yapılandırıcısıyla oluşturur ve işaret edeceği metodu ona parametre olarak verebilirsiniz.

Temsilci tanımı **delegate** anahtar kelimesiyle tanımlanır.

Örnek

delegate delegate_tipi delegate_adi (dönüş_tipi değişken_adi);

delegate string temsilci (int tms);

Bir temsilci, sınıf adından türetilmiş yeni bir metotla ya da sınıfın kendisiyle ilişkili **static** bir metotla kullanılabilir. Dikkat edilmesi gereken nokta, metottan geri dönen değer tipiyle temsilcinin tipinin uyuşmasıdır.

Temsilciler neden kullanılır?

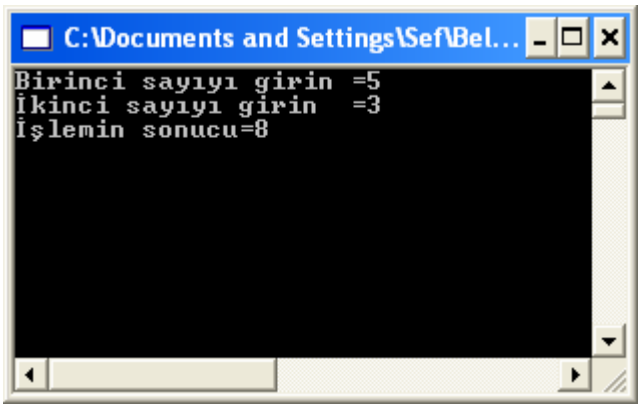
Temsilciler iki olaydan dolayı avantajlı oldukları için kullanılır. İlki olayları (events) desteklemesidir. İkincisi ise, bir metodun çalışma zamanında hangi metodun ya da metotların çalıştırılacağına karar vermesidir.

Örnek

Konsoldan girilen iki sayıyı tanımlanan temsilciye ve üye değişkene göre işleme tabi tutarak sonucu yine konsolda yazdıran örnektir.

```
using System;
namespace delege4
{
    class Class1
    {
        public static int Toplama(int x,int y)
        {
            return(x+y);
        }
        delegate int Temsilcim(int x,int y);

        static void Main(string[] args)
        {
            int a,b,islem;
            Console.Write("Birinci sayıyı girin =");
            a=int.Parse(Console.ReadLine());
            Console.Write("İkinci sayıyı girin  =");
            b=int.Parse(Console.ReadLine());
            Temsilcim t1=new Temsilcim(Toplama);
            islem=t1(a,b);
            Console.WriteLine("İşlemin sonucu="+islem.ToString());
            Console.Read();
        }
    }
}
```

Burada tanımlanan “Temsilcim” delegesiyle “t1” adlı üye değişken “Toplama()” metoduna göre tanımlanmaktadır. Girilen sayıları aslında “Toplama()” metodu yazmamasına rağmen “Toplama()” metoduna gönderip işleme tabi tutmakta ve sonucu “islem” değişkenine aktarmaktadır.



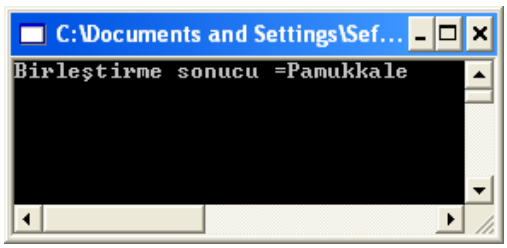
Resim 4.1: "delegate" kullanılan programın sonucu

Örnek

Programda verilen string iki bilgiyi temsilci aracılığıyla birleştiren program kodu şu şekildedir.

```
using System;
namespace delege6
{
    class Class1
    {
        public static string Birlestir(string a,string b)
        {
            return(a+b);
        }
        delegate string Temsilci(string a, string b);

        static void Main(string[] args)
        {   string sonuc;
            Temsilci Tms=null;
            Tms+=new Temsilci(Birlestir);
            sonuc=Tms("Pamuk","kale");
            Console.WriteLine("Birleştirme sonucu ="+sonuc);
            Console.Read();
        }
    }
}
```

"Temsilci" ismiyle oluşturulan delegeyle "Tms" üye değişkeni kendisine verilen "Pamuk" ve "kale" string bilgilerini "Birlestir()" metoduna göndererek birleştirilmelerini sağlamaktadır.



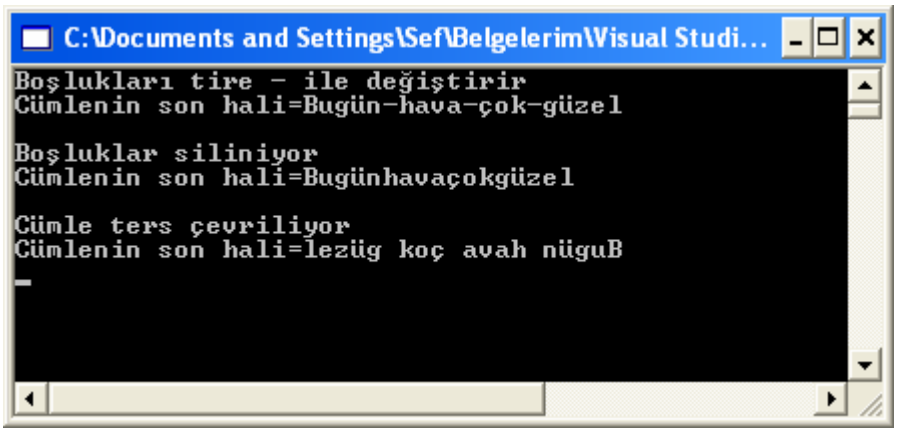
Resim 4.2: "delegate" ile string birleştirme örneğinin sonucu

Örnek

Programda verilen string bir bilgiyi temsilci aracılığıyla çeşitli string işlemlere tabi tutan programın kod satırlarıdır.

```
using System;
delegate string cumle(string c1);
class calistir
{
        static string boslukdegis(string islem)
        {
                Console.WriteLine("Boşlukları tire - ile değiştirir");
                return islem.Replace(' ','-');
        }
        static string bosluksil(string islem)
        {
                string gecici="";
                int k;
                Console.WriteLine("Boşluklar siliniyor");
                for(k=0;k<islem.Length;k++)
                        if(islem[k]!=' ')
                                gecici+=islem[k];
                return gecici;
        }
        static string terscevir(string islem)
        {
                string gecici="";
                int m,n;
                Console.WriteLine("Cümle ters çevriliyor");
                for(m=0,n=islem.Length-1;n>=0;n--,m++)
                        gecici+=islem[n];
                return gecici;
        }
        public static void Main()
        {
                cumle deg1=new cumle(boslukdegis);
                string c1;
                c1=deg1("Bugün hava çok güzel");
                Console.WriteLine("Cümlenin son hali="+c1);
                Console.WriteLine();

                deg1=new cumle(bosluksil);
                c1=deg1("Bugün hava çok güzel");
                Console.WriteLine("Cümlenin son hali="+c1);
                Console.WriteLine();

                deg1=new cumle(terscevir);
                c1=deg1("Bugün hava çok güzel");
                Console.WriteLine("Cümlenin son hali="+c1);
                Console.Read();
        }
}
```

Verilen cümlenin ilk olarak boşlukları tire işaretiyle değiştiriliyor. İkinci işlemde boşluklar silinip cümle boşluklar atılmış olarak ve üçüncü işlemde ise, cümle ters çevrilip yeniden yazdırılıyor.



Resim 4.3: "delegate" kullanılarak yapılan string işlem sonuçları

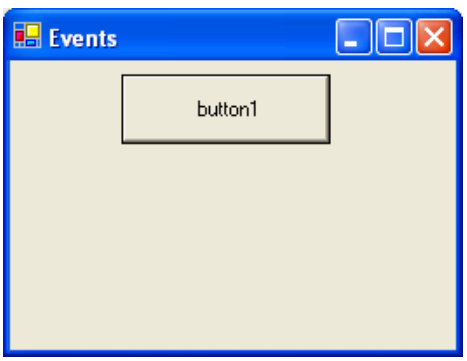
4.1. Olaylar (Events)

Olaylar bir sınıfın üyeleridir. Kullanımları kontrollerdeki şartın sağlanması durumunda istenilen faaliyetin çalıştırılması işine olay denir.

Olayla bağlantılı nesne, mevcut olay için bir EventHandler (Olay Yöneticisi) oluşturur. EventHandler'ın görevi bir nesnenin yaptığı işi diğer nesnelere aktarmaktır.

Örnek

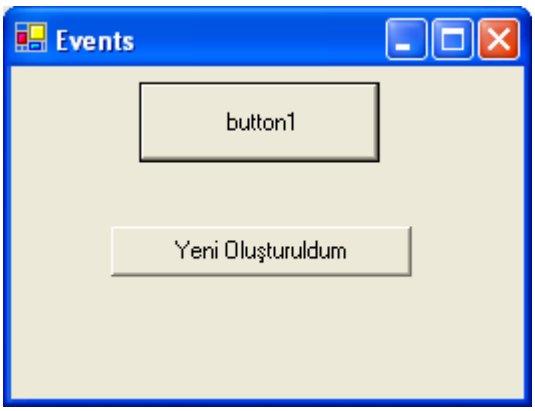
Form üzerindeki butona tıklandığında yeni bir buton oluşturulmaktadır. Yeni oluşturulan bu butona tıklandığında ise, buton adıyla birlikte işletilen metodun adını da mesaj kutusunda gösteren program kod satırları aşağıdaki gibidir.



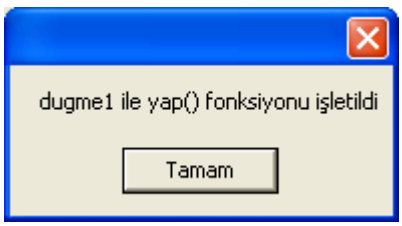
Resim 4.4: Örneğe göre formun tasarımı

```
using System;
using System.Windows.Forms;
namespace eventt1
{
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.Button button1;
        public Form1()
        {
            InitializeComponent();
        }
        public delegate void Temsilci(object sender,EventArgs e);
        public event Temsilci Click;
        public void yap(object gonder,EventArgs e)
        {
        MessageBox.Show(((Button)gonder).Name + " ile yap() "
            +"fonksiyonu işletildi");
        }
        private void button1_Click(object gonder, System.EventArgs e)
        {
            Button yenidugme=new Button();
            Controls.Add(yenidugme);
            yenidugme.Name="dugme1";
            yenidugme.Left=50;yenidugme.Top=80;
            yenidugme.Width=150;yenidugme.Height=25;
            yenidugme.Text="Yeni Oluşturuldum";
            Click=new Temsilci(yap);
            yenidugme.Click+=new EventHandler(Click);
        }
        Windows Form Designer generated code
        static void Main()
        {
            Application.Run(new Form1());
        }
    }
}
```

Formdaki buton üzerinde çift tıklandığında "button1_Click()" metodu açılır. Buton kontrolünden yeni bir buton tanımlanarak kontrollere eklenir ve form üzerindeki yeri belirtilir. Böylece form üzerine tanımlanan yeni düğme eklenmiş olur. Yeni eklenen düğmeye tıklandığında ise, yap() metodu işletilir.



Resim 4.5: Yeni bir butonun oluşturulmuş hali



Resim 4.6: Yeni oluşturulan butona tıklandığında yap() metodu işletilir

Örnek

Formu resim 4.7'deki aşağıdaki gibi tasarlayınız. Form üzerinde üç adet "Yazdır" butonu bulunmaktadır. Sadece birinci butona yapılacak işlemleri yazarak diğer iki butonun birinci buton gibi davranması sağlanır.

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

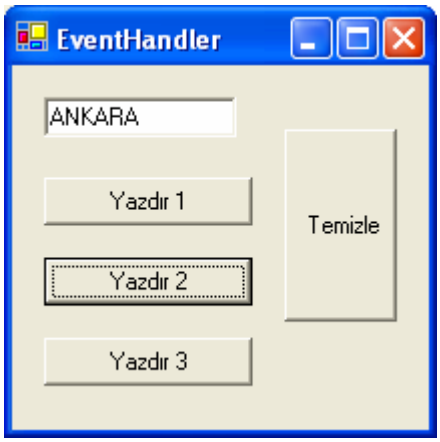
Birinci butonun yapacağı işlem yukarıdaki gibi yazıldıktan sonra ikinci ve üçüncü butonun aynı işlemi yapması aşağıdaki yöntemle sağlanır.

```
button2.Click += new System.EventHandler(button1_Click);
button3.Click += new System.EventHandler(button1_Click);
```

Formda bulunan dördüncü buton olan "Temizle" butonu metin kutusuna girilen bilgiyi silmektedir.

```
private void button4_Click(object sender, System.EventArgs e)
{
textBox1.Text="";
}
```

Böylece yapılan işlemleri formu çalıştırarak test edebilirsiniz.



Resim 4.7: EventHandler kullanılan form örneği

Çalışma anında istenilirse uygulanan bir EventHandler'ı devre dışı bırakabilirsiniz. Bunun için forma bir adet buton nesnesi ekleyiniz.

Resim 4.8: EventHandler olayı gerçekleştiren butonlardan istenileni çıkarmak için eklenen "EventHandler Çıkar" butonu

```
private void button5_Click(object sender, System.EventArgs e)
{
    button3.Click -= new EventHandler(button1_Click);
}
```

Eklenen buton nesnesine çift tıklayıp Click() metoduna yukarıdaki gibi kod satırlarını yazabilirsiniz. Burada üçüncü yazdır butonu devreden çıkarılmaktadır.

UYGULAMA FAALİYETİ-1

İşlem Basamakları	Öneriler
➢ Bir konsol uygulaması başlatınız.	
➢ Sınıf içinde int tipinde iki parametreye sahip ve geriye int tipinde değer gönderen delegate tanımlayınız.	➢ Delegate ismi "Temsilcim" olabilir.
➢ Yine aynı sınıf içinde dört işlemi yaptıracak dört adet metot tanımlayınız.	➢ Metot adları "Topla", "Cikar", "Carp", "Bol" olabilir.
➢ Bu metotlara parametre olarak delegate tanımında verdiğiniz parametreleri veriniz.	➢ Delegate tanımında parametre olarak (int m, int n) verdiyseniz burada da aynısını kullanınız.
➢ Her metodun içinde dört işlemi yaptıracak komutu yazınız.	➢ "return()" deyimiyle sonuç değerlerini "Main()" metoduna gönderiniz.
➢ Main() metodunda konsolu kullanarak iki sayı girişi yapıp ilgili değişkenlere aktarınız.	➢ Metotta verilen parametre tipleriyle konsoldan girilen değerlerin aktarılacağı değişken tipleri aynı olmalıdır. Gerekli satırlarda tip dönüşümleri yapılmalıdır.
➢ Tanımladığınız delegate adıyla bir delegate üye değişkeni oluşturunuz.	➢ Tanımladığınız delegate isminin kısaltılmış hali olabilir.
➢ Delegate üye değişkeninin hangi metotları temsil edeceğini belirtiniz.	➢ Dört metodun isimlerini yazınız.
➢ Metotlarda yapılan işlemlerin geriye dönen sonucunu yazdırmak için bir değişken tanımlayınız ve sonuçları bu değişkene aktarınız.	➢ "sonuc=Tms(sayi1,sayi2);" şeklinde bir kod yazabilirsiniz.
➢ Programı çalıştırınız. Çalışma esnasında hata oluşmuşsa kod satırlarına dönerek yazım hatalarınızı kontrol edip tekrar çalıştırınız.	➢ Amacınızı, kod satırlarını ve işlem sonucunun ekran görüntüsünü defterinize yazınız.

UYGULAMA FAALİYETİ-2

Aşağıda verilen soruları ödev olarak yapınız. Sonuçları rapor halinde öğretmeninize sununuz.

- Console'dan girilen rakama göre haftanın hangi günü olduğunu ve gün adının kısaltılmış halini temsilci kullanarak yapınız.

- Form üzerine 5 adet metin kutusu yerleştiriniz. Birinci metin kutusuna bilgi girildikten sonra klavyeden Enter tuşuna basıldığında imleci bir sonraki metin kutusuna geçiren ve diğer metin kutuları içinde aynı işlemi yaptıran programı yapınız.

- Console'dan string bir cümle girerek bu cümle içindeki boşluk karakterlerinin ve diğer karakterlerin sayısını ayrı ayrı bulduran programı delegate ile yapınız.

ÖLÇME VE DEĞERLENDİRME

A. OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki soruları dikkatlice okuyarak doğru/yanlış seçenekli sorularda uygun harfleri yuvarlak içine alınız. Seçenekli sorularda ise uygun şıkkı işaretleyiniz. Boşlukları uygun şekilde doldurunuz.

1. Bir veya birden fazla metodu program içinde temsil eden referans türü nesneleri tanımlamak için …….kullanılır.

2. Temsilciler metotların içindeki işlemleri gerçekleştirmek için oluşturulur. (D/Y)

3. Metot içinden dönen değerin tipiyle temsilcinin tipinin aynı olması gerekmez.(D/Y)

4. Çalışma zamanında hangi metodun çalıştırılacağına **derleyici / delege** karar verir.

5. Bir nesnenin yaptığı işi diğer nesnelere aktaran komut aşağıdakilerden hangisidir?
A) Event
B) Object
C) Delegate
D) EventHandler

DEĞERLENDİRME

Cevaplarınızı cevap anahtarı ile karşılaştırınız. Doğru cevap sayınızı belirleyerek kendinizi değerlendiriniz. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt yaşadığınız sorularla ilgili konulara geri dönerek tekrar inceleyiniz. Tüm sorulara doğru cevap verdiyseniz diğer öğrenme faaliyetine geçiniz.

MODÜL DEĞERLENDİRME

PERFORMANS TESTİ (YETERLİK ÖLÇME)

Modül ile kazandığınız yeterliği aşağıdaki ölçütlere göre değerlendiriniz.

DEĞERLENDİRME ÖLÇÜTLERİ	Evet	Hayır
➢ İki ayrı metin kutusunda iki farklı sayı girdiniz mi?		
➢ Sayıların değişken türlerini sınırlarına göre seçtiniz mi?		
➢ Değişkenlerle işlem yaptınız mı?		
➢ İşlem sonuçlarını mesaj kutusunda yazdırdınız mı?		
➢ Sayı dizisi tanımladınız mı?		
➢ Diziye sayı girişi yaptınız mı?		
➢ Dizi işlemleri için döngü çeşidini belirlediniz mi?		
➢ Dizi elemanlarını topladınız mı?		
➢ InputBox ()'ı kullandınız mı?		
➢ Sayıları convert ettiniz mi?		
➢ Diziye girilen sayıların toplamını hesaplatma ve mesaj kutusunda yazdırdınız mı?		
➢ Konsol uygulaması başlattınız mı?		
➢ Sınıf tanımladınız mı?		
➢ Sınıf türettiniz mi?		
➢ Sınıf içinde değişken kullandınız mı?		
➢ Main() metodunda bilgi yazdırdınız mı?		
➢ Sınıflarda akış kontrol deyimlerini kullandınız mı?		
➢ Üye değişken tanımladınız mı?		
➢ Üye değişken kullandınız mı?		
➢ Arayüz tanımladınız mı?		
➢ Delege tanımladınız mı?		
➢ Metot tanımladınız mı?		
➢ Delege üye değişkeni oluşturdunuz mu?		
➢ Delegenin temsil ettiği metotları belirlediniz mi?		
➢ Kod satırlarını derleyip çalıştırdınız mı?		

DEĞERLENDİRME

Yaptığınız değerlendirme sonucunda eksikleriniz varsa öğrenme faaliyetlerini tekrarlayınız.

Modülü tamamladınız, tebrik ederiz. Öğretmeniniz size çeşitli ölçme araçları uygulayacaktır, öğretmeninizle iletişime geçiniz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ-1 CEVAP ANAHTARI

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

ÖĞRENME FAALİYETİ-2 CEVAP ANAHTARI

1	Doğru
2	Yanlış
3	[,]
4	Düzensiz
5	C
6	C

ÖĞRENME FAALİYETİ–3 CEVAP ANAHTARI

1	Doğru
2	A
3	A
4	Yanlış
5	Doğru
6	public

ÖĞRENME FAALİYETİ–4 CEVAP ANAHTARI

1	delegate
2	Doğru
3	Yanlış
4	delege
5	D

KAYNAKÇA

- SCHILDT Herbert, **Herkes için C#**, Alfa Basım Yayım Dağıtım Ltd.Şti, İstanbul 2005.
- DEMİRLİ Nihat, İNAN Yüksel, **Visual C#.NET 2005**, Palme Yayıncılık, Ankara 2006.
- Karagülle İHSAN, **Visual C#.Net Başlangıç Rehberi**, Türkmen Kitabevi, İstanbul 2004.
- YANIK Memik, **Microsoft Visual C#.NET,** Seçkin Yayıncılık, Ankara 2004.
- ZENGİN Abdullah, **C# 2005**, Nirvana Yayınları, Ankara 2006.
- ZEKİ Yasemin, **Adım Adım C++ Uygulamaları**, Nirvana Yayınları, Ankara 2006.
- www.csharpnedir.com.
- www.msakademik.net