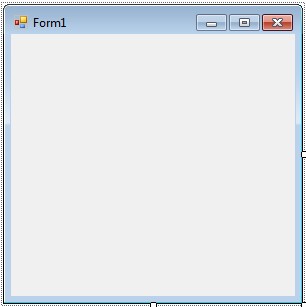
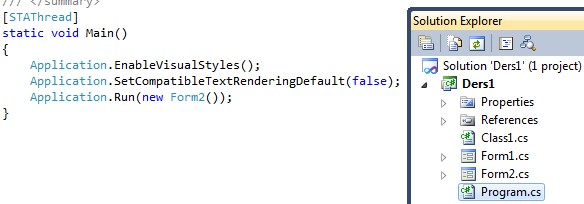
# Pencereler (Forms):

Form nesneleri, Windows tabanlı programlarda gördüğümüz arabirimlerdir ve diğer tüm nesneleri üzerlerinde tutarlar. Her form aynı zamanda bir sınıftır (class) ve bu formun birkaç örneği oluşturulabilir yâda başka bir formdan miras (inherit) de alabilir. Windows formlarına ilişkin sınıflar ***System.Windows.Forms*** isim alanında (namespace) bulunmaktadır. Program çalıştırıldığında Form sınıfı, ekranda boş bir pencerenin gösterilmesini sağlar ve form özelliklerini belirler. Form sınıflarının birçok özelliği, metodu ve olayı vardır.

***Yeni Form oluşturma :***

Yeni bir Windows uygulaması oluşturulduğunda “form1” adında bir form varsayılan olarak uygulamaya eklenir. Tasarım ekranındaki form1’e, *ToolBox* (araç kutusu) ile kontroller, menüler ve görsel öğeler eklenebilir. Projeye yeni bir form eklemek için *“Project”* menüsünden *“Add Windows Forms…”* seçeneği kullanılabilir.

Projede birden fazla form varsa ilk açılacak form yani ana formun ismi ***Solution Explorer*** kısmındaki Program.cs içerisinde bulunan ***Main()*** metodu içinde yazılmalıdır.

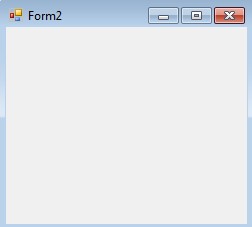
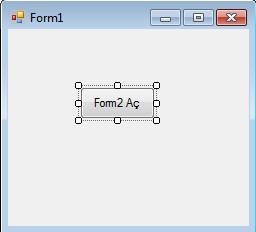


Yukarıda *Application.Run(new Form2());* satırında Form2 yazıldığından bu form ana form olarak çalışacaktır.

***Formlar Arası Geçiş:***

Uygulamalarda bir formdan başka bir formun açılması ve ayarlanması sık karşılaşılan bir durumdur. Gösterilmesi istenilen form nesnesi çağıracak olan formda tanımlanmalıdır. Örneğin; Form1 den Form2 gösterilecekse Form1 in kodlarına ***Form2 YeniForm = new Form2();*** satırı eklenmeli ve *Show()* yada

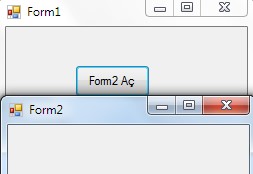
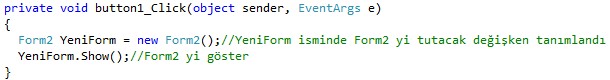
S*howDialog()* metotları kullanılmalıdır. *ShowDialog* metodu, form gösterildikten sonra kapanana kadar diğer formlara erişimi engeller.



## Örnek Uygulama:

Form1 ve Form2 isminde iki tane form oluşturunuz ve ana form varsayılan olarak Form1 olsun. Form1 üzerine bir tane düğme

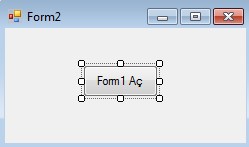
(button1) nesnesi yerleştiriniz ve *text* özelliğini “Form2 Aç” yapınız. Düğmenin *Click* olayına aşağıdaki kodları yazınız. Proje çalıştırıldığında ilk olarak Form1 açılacaktır. Düğmeye tıklandığında ise Form2 açılacaktır.



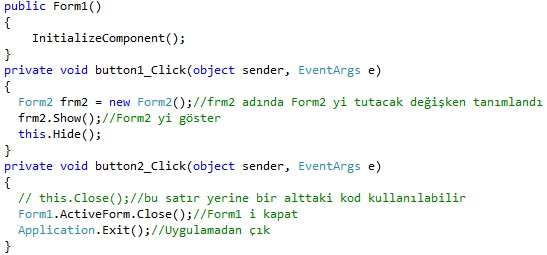
Aynı örneği ***ShowDialog()*** metodu ile yapınız ve ***Show()*** metodu ile arasındaki farkı görünüz.

## Örnek Uygulama:

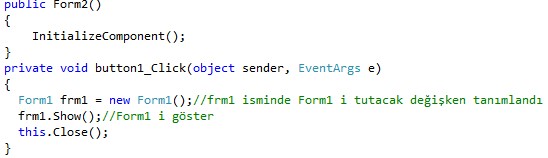
Yukarıdaki örneğe göre; Form1 den Form2 gösterildiğinde Form1 gizlensin, aynı şekilde Form2 den Form1 gösterildiğinde Form2 gizlensin.



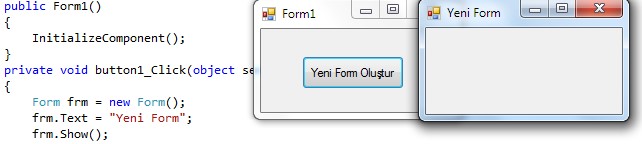
Form1 de yazılacak kodlar aşağıdadır.



Form2 de yazılacak kodlar aşağıdadır.



***Kod ile Yeni Form Oluşturmak:***



Aşağıda Formların bazı önemli özellikleri gösterilmiştir.

**Form Özellikleri (Properties)**

***Name :***

Formun program kodu kısmında kullanılacak ismidir ve küçük/büyük harf ayrımı vardır.

***AcceptButton :***

Form üzerinde ENTER tuşuna basıldığında formdaki hangi düğmenin işlem yapması gerektiğini belirler. Bu özellikte form üzerinde var olan butonlardan biri seçilir.

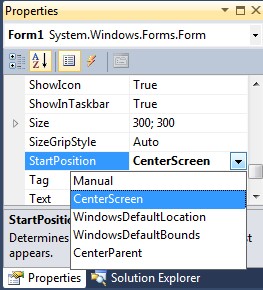
***AllowDrop :***

Bu özellik ile formun üzerine mouse ile sürükle-bırak işlemi yapılıp yapılamayacağı ayarlanır.

***AutoScaleMode :***

Ekran çözünürlüğü değiştiği zaman, formun veya kontrolün hangi özelliğinin bu çözünürlüğe göre ayarlanacağını belirtir.

***AutoScroll :***

Bu özellik ile, formun içindeki nesneler, ekranın görünebilir alanından daha büyük ise otomatik olarak bir *scrollbar* eklenip eklenmeyeceğini ayarlamaya yarar.

***StartPosition :***

Formun ekranda gösterilme pozisyonunu yani formun çalışma pozisyonu belirler.

*Manual:* Form, konum (Location) özelliğinde belirtilen koordinatlar da ekrana gelir.

*Center Screen:* Ekrana ortalanmış olarak gelir.

*WindowsDefaultLocation:* Windows’un mevcut koordinatlarına göre ekrana gelir.

*WindowsDefaultBounds:* Var olan koordinatlar ve mevcut ekran çözünürlüğüne göre ekrana gelir.

*CenterParent:* Ana formun ortasına denk gelecek şekilde ekrana gelir.

***BackColor :***

Formun arkaplan (zemin) rengini belirler.

***BackGroundImage :***

Forma yerleştirilecek resim (image) bu özellikten seçilir.

***BackGroundImageLayout :***

Forma yerleştirilen resmin görüntülenme şeklini belirler.

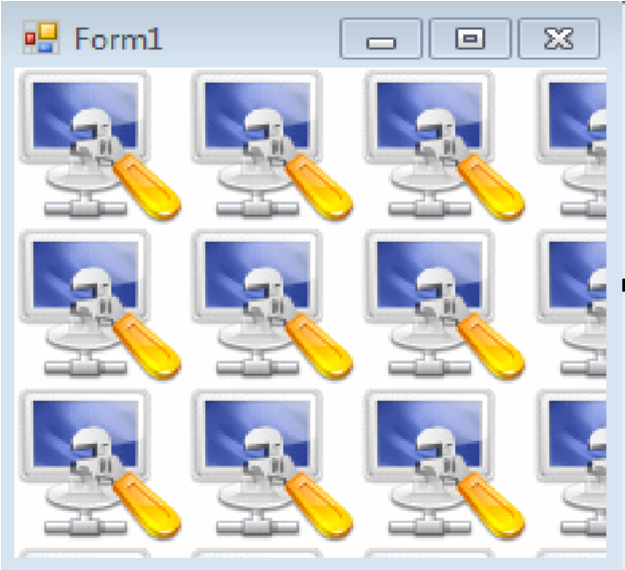
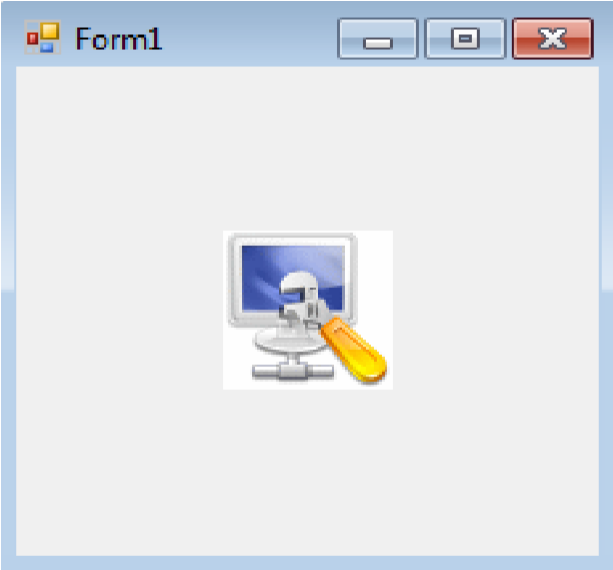
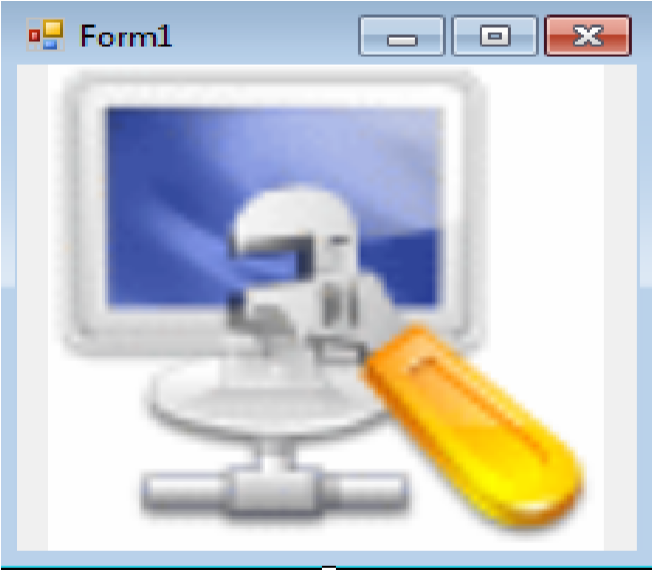
*None:* Resim olduğu gibi gösterilir.

*Tile:* resim birden fazla parça şeklinde gösterilir.

*Center:* Resim formun ortasında gösterilir.

*Stretch:* Resim formun tamamına yayılarak gösterilir.

*Zoom:* Resmi büyüterek gösterir.



Yukarıda formun arka planı olarak seçilen resme *BackGroundImageLayout* özelliği sırasıyla *Zoom*, *Center* ve *Tile* seçilmiştir.

***ContextMenuStrip :***

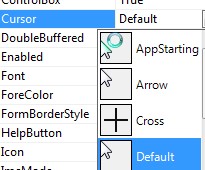
Formun üzerinde sağ tıklayınca gösterilecek menünün seçildiği özelliktir. İlerideki konularda anlatılacaktır.

***ControlBox :***

Formun sağ üst köşesindeki  simgelerinin görünür olup olmamasını sağlar varsayılan değeri *true* dır.

***Cursor :***

Formun üzerinde fare işaretçinin alacağı görünümü belirler.



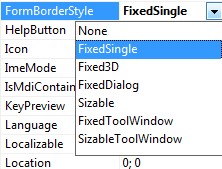
***Enabled :***

Formun aktif olup olmamasını belirler varsayılan değeri *true* yani aktiftir.

***ForeColor :***

Form üzerine yerleştirilen kontrollerin yazı renklerini belirler.

***FormBorderStyle :***

Formun kenar sitili ayarlarını yapan özelliktir. Varsayılan değeri *Sizable* dir yani formun boyutları kullanıcı tarafından değiştirilebilir.

*FixedSingle:* Formun kenarlarından boyutları değiştirilemez.

*Fixed3D:* Formun kenarlarından boyutları değiştirilemez ve formu 3 boyutlu gösterir.

*FixedToolWindow:* Formu araç kutusu penceresi gibi gösterir ve form boyutları değiştirilemez.

*SizableToolWindow*: bir önceki seçenek gibi davranır tek fark formun boyutları değiştirilebilir.

***Icon :***

Formun sol üst köşesindeki  ikon resmini değiştirir. Resim uzantısı \*.ico dur.

***KeyPreview :***

Form üzerinde hangi tuşa basıldığını algılar. Varsayılan değeri *False* dır.

***Location :***

Formun ekranda soldan (X) ve üstten (Y) kaç piksel konumunda olacağını ayarlar. Formun belirtilen X ve Y konumunda gösterilmesi için *StartPosition* özelliğinin *Manual* olması gerekir.

***Locked :***

Bu özellik Formun hem *Location* olarak hem de Formun *Drop* yani sürükleme özelliğini değiştirilemez kılar.

***MainMenuStrip :***

Formda gösterilecek ana menüyü ayarlayan özelliktir. İlerideki konularda anlatılacaktır.

***MaximizeBox :***

Formun sağ üst köşesinde gösterilen  büyült (ekranı kapla) düğmesinin görünüp görünmeyeceğini ayarlar.

***MaximumSize :***

Formun alabileceği maksimum büyüklüğü ayarlar. Kullanıcı formu ne kadar büyültmek isterse istesin verilen değerden büyük bir değer alamaz.

***MinimizeBox :***

Formun sağ üst köşesinde yer alan  küçült düğmesinin görünüp görünmeyeceğini ayarlar.

***MinimumSize:***

Formun alabileceği en küçük boyut değerini ayarlar.

***Opacity :***

Formun saydamlığını ayarlar. Yüzde(%) bir değer ister. 100% tam görünür, 0% görünmez anlamına gelir.

***Padding :***

Forma eklenen kontroller arasında ne kadar boşluk olması gerektiğini ayarlar

***ShowIcon :***

Formun başlık çubuğunda ikon 'un gösterilip gösterilmeyeceğini ayarlar.

***ShowInTaskBar :***

Uygulamanın, görev çubuğunda görünüp görünmeyeceğini ayarlar.

***Size :***

Formun boyutlarını piksel biriminde ayarlar.

***Text :***

Formun sol üstte bulunan ikonun yanındaki metini değiştirir ve bu değer varsayılan olarak *Form1* dir. Burada dikkat edilmesi gereken *Text* özelliği formun adını değiştirmez.

***TopMost :***

Formun diğer formların en üstünde olmasını sağlar ve form kapatılmadan diğer formlara geçilemez.

***WindowsState :***

Formun ilk gösterildiğinde tam ekran veya normal boyutlarında mı gösterileceğini ayarlar. *Normal*, *Minimized*, *Maximized* özellikleri mevcuttur.

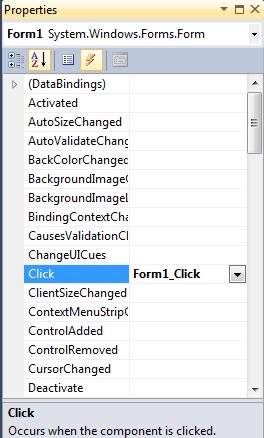
**Form Olayları (Events)**

Form üzerindeki iken fare ve klavye ile yapılan olaylara program kodu yazılabilir. Aşağıda önemli olan form olayları anlatılmıştır:

***Click :***

Form üzerinde farenin tıklanması ile oluşur.

***FormClosing :***

Formun sağ üst köşesindeki simgesi ile form kapanmadan önce oluşur.

***FormClosed :***

Form kapandıktan sonra oluşur.

***Load :***

Form yüklenirken yani gösterilirken bu olay meydana gelir.

***KeyDown :***

Form üzerinde iken bir tuşa basılması ile oluşur.

***KeyUp :***

Form üzerinde iken basılan tuşun bırakılması ile oluşur.

**Form Metotları (Methods)**

Formlara uygulanabilecek metotlar, özelliklere ek olarak kullanılabilir.

***Hide :***

Formu gizler.

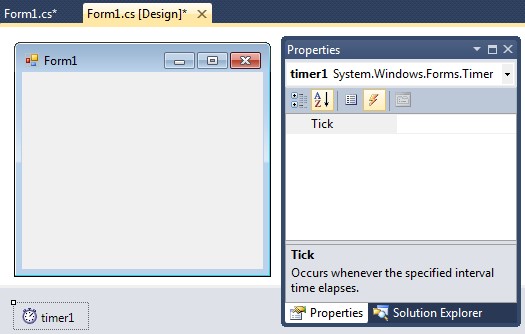
***Close :***

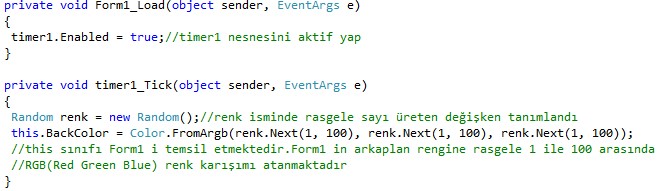
Formu kapatır.

***Show :***

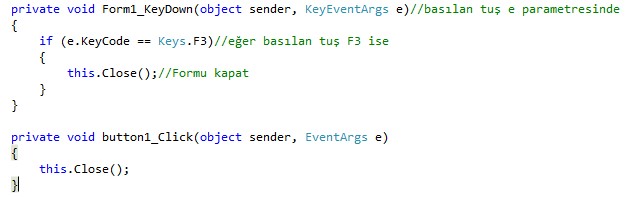
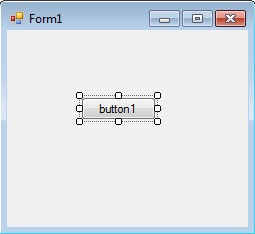
Formu gösterir.

## Örnek Uygulama:

Yanda gösterildiği gibi boş bir form açarak, Araç Kutusundan (Toolbox) bir *timer* nesnesini oluşturunuz. Daha sonra Form1 nesnesinin *Load* olayında timer1 nesnesini aktif hale getiriniz. Sonrada *timer1* nesnesinin *Tick* olayını tıklayarak aşağıdaki kodları yazınız.



Burada *timer1\_Tick* olayında önce rastgele sayı üreten renk isminde bir değişken tanımlandı sonrada *renk.Next(1,100)* ifadesi ile 1 ile 100 arasında kırmızı,yeşil ve mavi renk karışımını üretilerek formun arka plan rengi değiştirildi.



***Örnek Uygulama:***

Yanda

gösterildiği

gibi

b

oş

bir

form

açarak

forma

bir

buton

nesnesi

koyunuz.

Daha

sonra

düğmenin

*Click*

ve

formun

*KeyDow*

*n*

olayına

kodları

yazınız.

Bu

örnekte

aşağıdaki

tıklandığında

yada

düğmeye

*F3*

tuşuna basıldığında

form

kapatı

lır

.

**Mesaj**

**Pencereler**

**i**

**(**

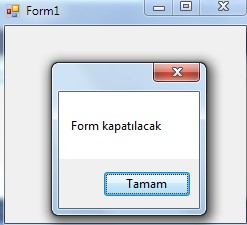
**MessageBox**

**):**

Kullanıcıya bilgi, uyarı, onay yada hata mesajı vermek için kullanılan sınıftır. *MessageBox* 4 şekilde yazılabilir.

**1.** ***MessageBox.Show ( “Açıklama”);***

## Örnek Uygulama:



Formun

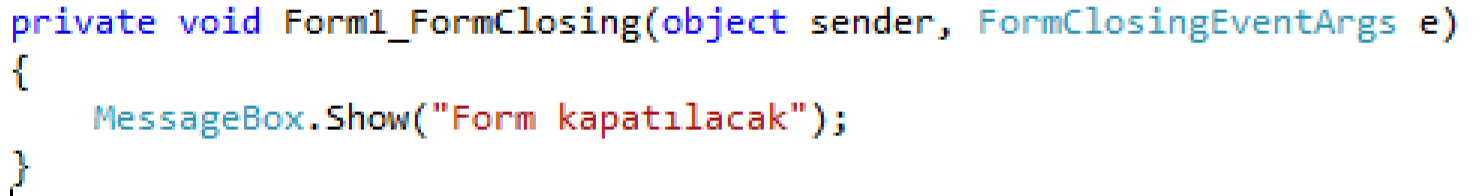
*Closing*

olayına aşağıdaki kodları yazınız ve

proje

çalıştırı

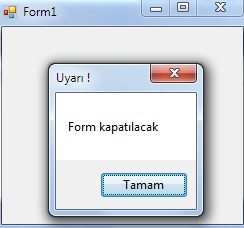
ldığında formun kapatma düğmesine basınız.

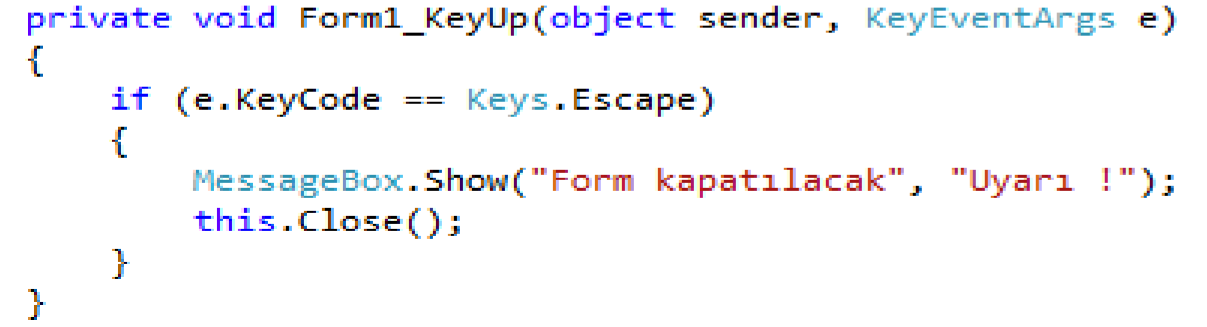


**2.** ***MessageBox.Show ( “Açıklama”,”Pencere başlığı”);***

## Örnek Uygulama:

Formun *KeyUp* olayına aşağıdaki kodları yazınız ve projeyi çalıştırdıktan sonra klavyede ESC tuşuna basınız.

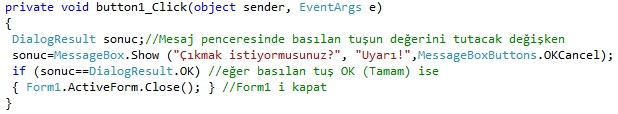
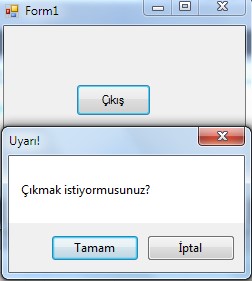




**3. *MessageBox.Show ( “Açıklama”,”Pencere başlığı”,Düğmeler);*** Mesaj penceresinde kullanıcının işlem yapması için düğmelerin kullanıldığı mesaj penceresidir. Yani varsayılan olarak gelen “Tamam” düğmesinin yerine “Evet-Hayır-İptal” gibi düğmelerin kullanabildiği mesaj pencereleridir. Düğmeleri göstermek için *MessageBoxButtons* ifadesi kullanılır.

## Örnek Uygulama:

Form üzerine şekilde görüldüğü gibi bir düğme koyunuz daha sonra düğmenin *Click* olayına aşağıdaki kodları yazınız.



Mesaj

penceresinde

*“Tamam”*

düğmesine

tıklandığında

*sonuc*

değişkeninde

*OK*

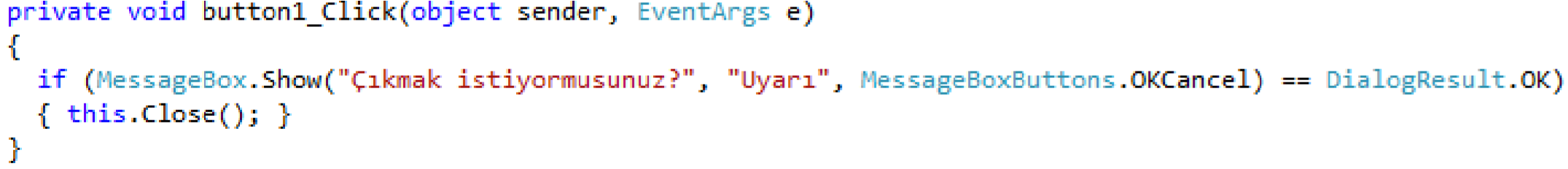
olacaktır ve

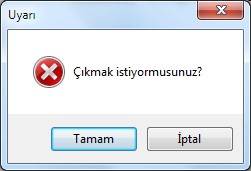
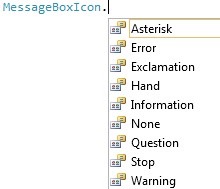
*if*

komutu çalışacaktır.

***Örnek Uygulama:***

Yukarıdaki örneğin başka bir yolu aşağıda gösterilmiştir.





**4**

**.**

***MessageBox***

***.Show***

***“Açıklama”,”Pencere başlığı”***

***(***

***,Düğmeler,Mesaj simgesi***

***;***

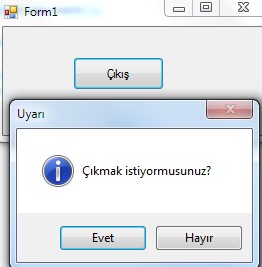
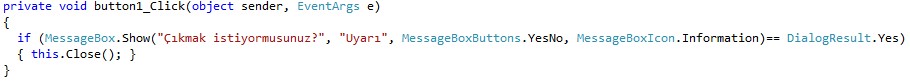
***)***

Mesaj penceresinde gösterilecek simge

*MessageBoxIcon*

ifadesi ile

seçilir.



***Örnek Uygulama:***

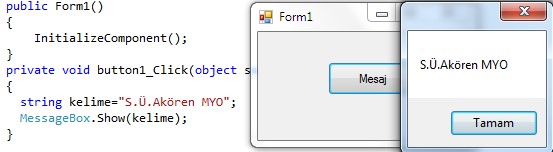
## Örnek Uygulama:

Mesaj penceresi gösterildiğinde varsayılan olarak birinci düğme seçili olarak gösterilir. Mesaj penceresinde seçili olan düğme *MessageBoxDefaultButton* ifadesi ile belirlenir.



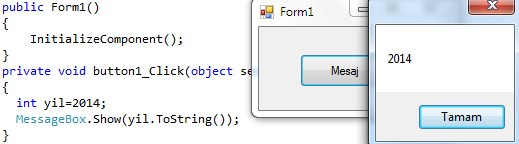
## Örnek Uygulama:

Aşağıda önce bir değişken tanımlanmış ve mesaj penceresinde bu değişken gösterilmiştir.



## Örnek Uygulama:

Aşağıdaki uygulamada *yil* değişkeni tamsayı değişken olduğu için *ToString()* ile karaktere çevrilmiştir.



## Örnek Uygulama:



# NumericUpDown Kontrolü:

Sayısal değerleri yukarı-aşağı olarak artıran yada azaltan kontroldür. Kullanıcı isterse değerleri oklarla artırıp azaltmadan da değer girebilir.

***NumericUpDown Özellikleri:***

***Increment :***

Aşağı yukarı oklar ile sayıların artma ve azalma adımlarını belirler.

***Maximum :***

Kontrolde gösterilecek sayının en büyük değerini belirler.

***Minimum :***

Kontrolde gösterilecek sayının en küçük değerini belirler.

***ThousandSeparators :***

Sayıların basamak ayracının gösterilmesini sağlar.

***Value :***

Kontrolde o andaki sayı değerini gösterir.

***ReadOnly :***

*True* değerini alırsa kullanıcı sayı girişi yapamaz.

***NumericUpDown Olayları :***

***ValueChanged :***

Gösterilen sayı değiştiğinde bu olay meydana gelir.

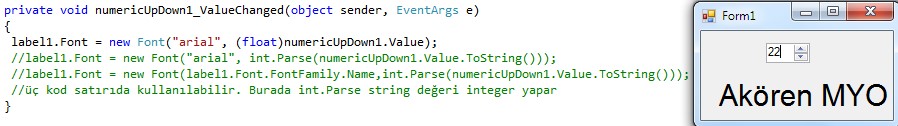
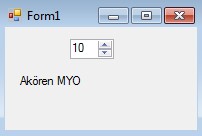
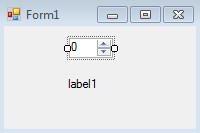
***NumericUpDown Metotları :***

***UpButton :***

Yukarı düğmesine basar ve sayı değerinin artırılmasını sağlar.

***DownButton :***

Aşağı düğmesine basar ve sayı değerinin azaltılmasını sağlar.



***Örnek Uygulama:***

Form

üzerine

bir

tane

*numericUpDown*

ve

bir

tanede

*label*

kontrolü

koyunuz.

*numericUpDown*

kontrolünün

*value*

değerini

10

yapınız

ve

*label*

kontrolünün

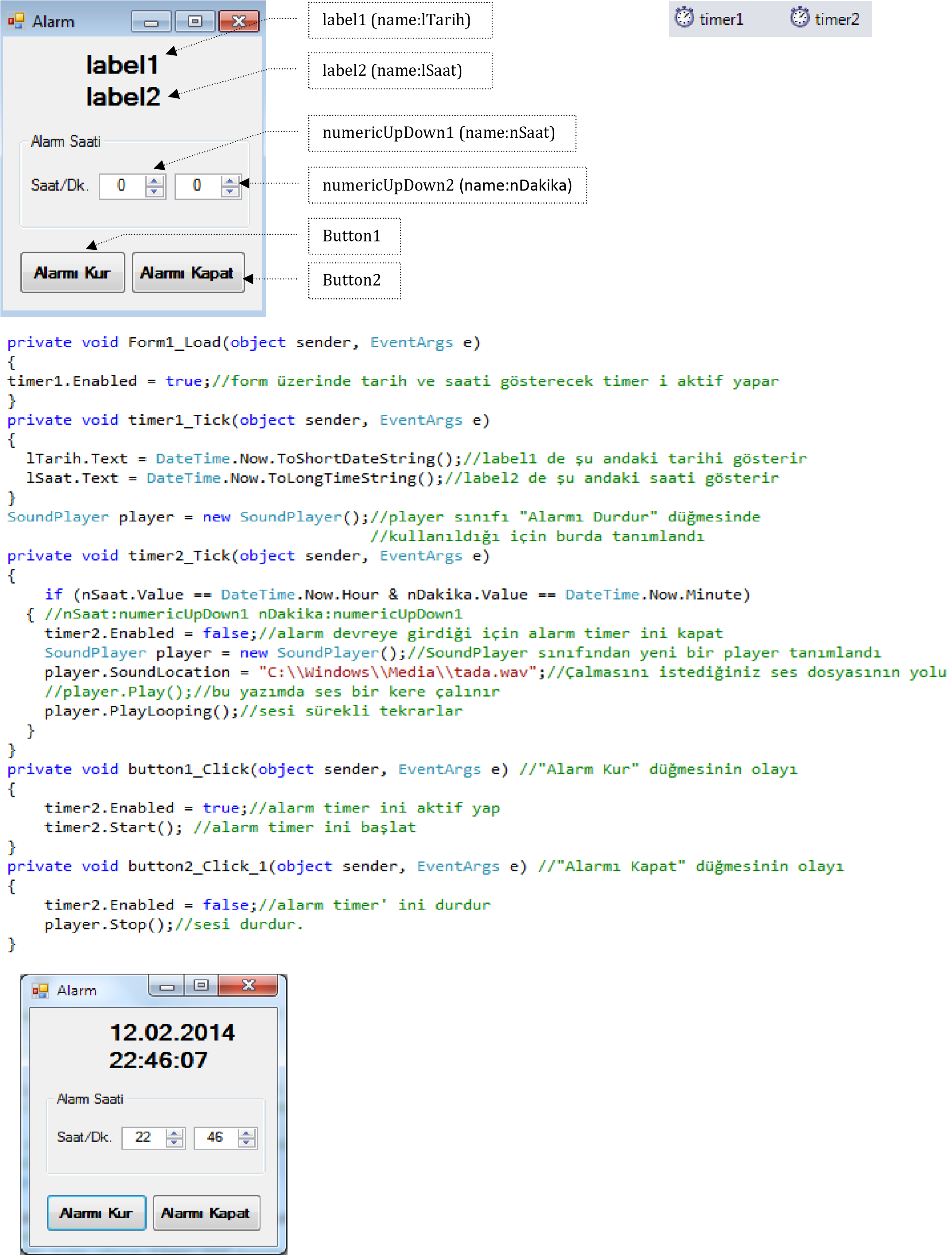
*text*

özelliğine “Akören MYO” giriniz.

## Örnek Uygulama:

Seçilen tarih ve saate göre alarm sesi çıkaran uygulama. Aşağıdaki formda gösterilen kontrolleri oluşturunuz.

*SoundPlayer* sınıfını kullanabilmek için kütüphane satırına *Using System.Media* satırının eklenmesi gerekir.



# DomainUpDown Kontrolü:

*NumericUpDown* kontrolü ile aynı yapıdadır ancak sayısal değerler yerine *string* tipinde değerler tutar. Bu *string* değerler *domainUpDown* kontrolünün *Items* özelliğinden girilir

***DomainUpDown Özellikleri:***

***Items :***

Kontrol içinde liste şeklinde gösterilecek *string* ifadeler bu özellik ile girilir.

***SelectedItem :***

Kontrolde listeden seçilen öğeyi gösterir.

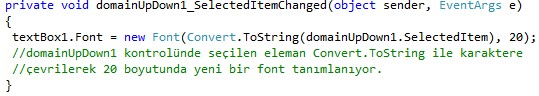
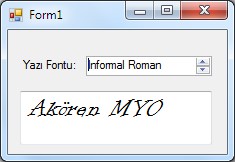
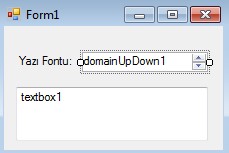
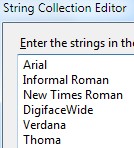
***Wrap :***

Kontrolde liste sonuna gelindiğinde baştaki veya sondaki öğeye geri dönülmesini belirler.

***DomainUpDown Olayları :***

***SelectedItemChanged :***

Kontrolde gösterilen listedeki seçili olan öğe değiştiğinde meydana gelir.



***Örnek Uygulama:***

Yanda

form

tasarımında

gösterildiği

gibi

kontrolleri

form

üzerine

yerleştiriniz.

Formda

*domainUPDown*

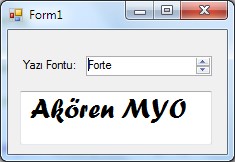
kontrolünün

*Items*

özelliğine

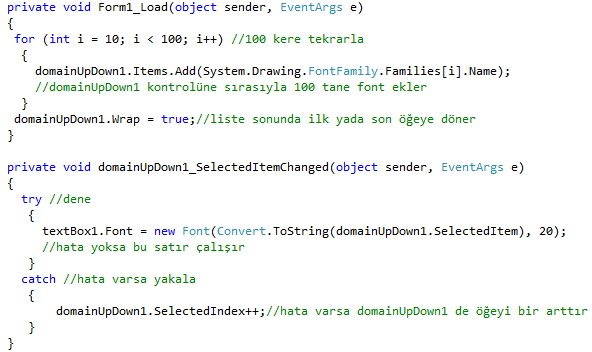
aşağıdaki yazı fontlarını yazınız.

***Örnek Uygulama:***

Yukarıdaki uygulamayı *domainUPDown* kontrolünde yazı fontlarını sistemden alacak şekilde değiştirelim.

*try-catch* yapısını kullanmadan uygulama çalıştırılırsa bazı fontlarda hata verme olasılığı vardır. Bu nedenle *try-catch* yapısı kullanılarak olası hatalar engellenmişti. *domainUpDown1.SelectedIndex++* ifadesi *domainUpDown* kontrolündeki öğeyi 1 arttırır.

Yani *domainUpDown1.SelectedIndex++* ifadesi ile *domainUpDown1.SelectedIndex= domainUpDown1.SelectedIndex+1* ifadesi aynıdır.



# dateTimePicker Kontrolü:

Form üzerinde takvim gösteren kontroldür ve seçilen tarih *date* tipindedir.



***DateTimePicker Özellikleri :***

***Format :***

Tarihin gösterilme biçimini belirler. *Long* özelliği uzun tarih biçimini, *Short* ise kısa tarih biçimini belirler. *Time* sadece zamanı gösterir. *Custom* özelliği, tarihin *CustomFormat* özelliğine girilen formatta gösterileceğini belirler.

***Value :***

*dateTimePicker* kontrolünde seçilen tarih değerini belirler.

***ShowUpDown :*** *numericUpDown* kontrolünde olduğu gibi aşağı ve yukarı oklar gösterilir. 

## Örnek Uygulama:

Form üzerine bir adet *dateTimePicker* kontrolü koyunuz, formun *Load* olayı aşağıdadır. Dikkat edilirse tarih kodda yazılan biçimde gösterilmektedir.

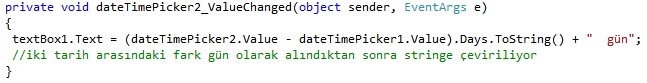
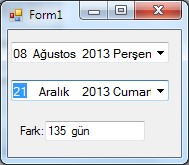
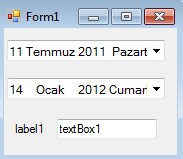


Aynı uygulamayı kod ile değil de, *dateTimePicker* kontrolünün özelliklerinden yapınız.

## Örnek Uygulama:

Form üzerinde seçilen iki tarih arasındaki farkı bulan uygulama görüldüğü gibi *dateTimePicker2* kontrolünün

*ValueChange* olayına yazılmıştır. Yani *dateTimePicker2* de tarih değiştiğinde iki tarih arasındaki fark



hesaplanacaktır.

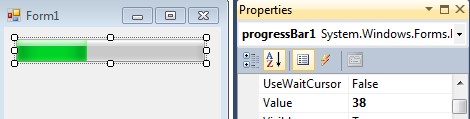
# ProgressBar Kontrolü:

Herhangi bir işlemin ilerleyişini gösteren kontroldür.

***ProgressBar Özellikleri :***

***Value :***

İşlemin o anda % kaçta olduğunu gösteren pozisyon değeridir.



***Minimum :***

Kontrolün alabileceği en küçük değeri belirler ve varsayılan değeri 0 dır.

***Maximum :***

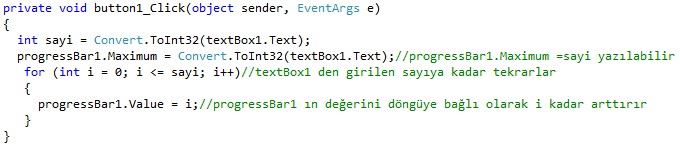
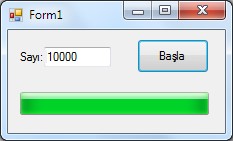
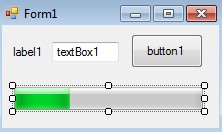
Kontrolün alabileceği en büyük değeri belirler ve varsayılan değeri 100 dür.

***Step :***

Değer artış miktarıdır.

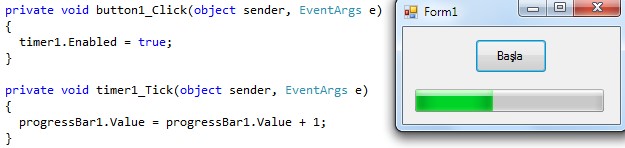
***Style :***

*ProgressBar* kontrolünün görünüm sitilini belirler.



***Örnek Uygulama:***

## Örnek Uygulama:



Aşağıdaki gibi bir form tasarlayınız ve

*button1*

,

*progressBar1*

ve

*timer1*

kontrolünü form üzerine yerleştiriniz.

Burada

başla

düğmesine

tıklandığında

*timer1*

aktif

oluyor

ve

*timer1*

nesnesinin

*Tick*

olayı

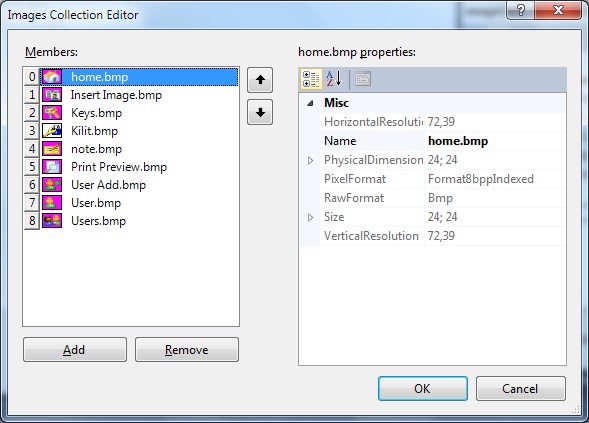
devreye

giriyor.

Burada her 1 saniyede *progressBar1* in değeri bir arttırılıyor.

# Imagelist Kontrolü:

Resimleri liste şeklinde tutarak kontrollerde gösterilmesini sağlar. 

*ImageList* de tutulan resimleri diğer kontrollerde göstermek için bu kontrolün *ImageList* özelliği ile yapılır. *ImageList* kontrolü daha çok *Listview* ve *TreeView* gibi birden fazla resmin gösterileceği kontrollerde kullanılır.

***Imagelist Özellikleri :***

***Images :***

Liste şeklinde tutulacak resimlerin seçildiği özelliktir. *Images* özelliğinde açılan pencereden

“Add” ile istenilen resimler seçilir.

***ImageSize :***

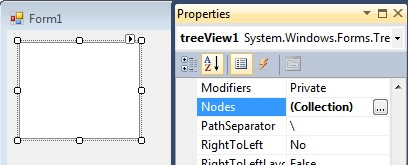
Resimlerin boyutlarını gösterir varsayılan değer 16\*16 dır.

***TransparentColor :***

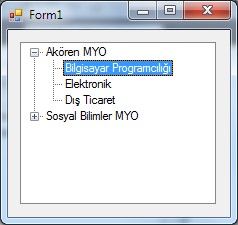
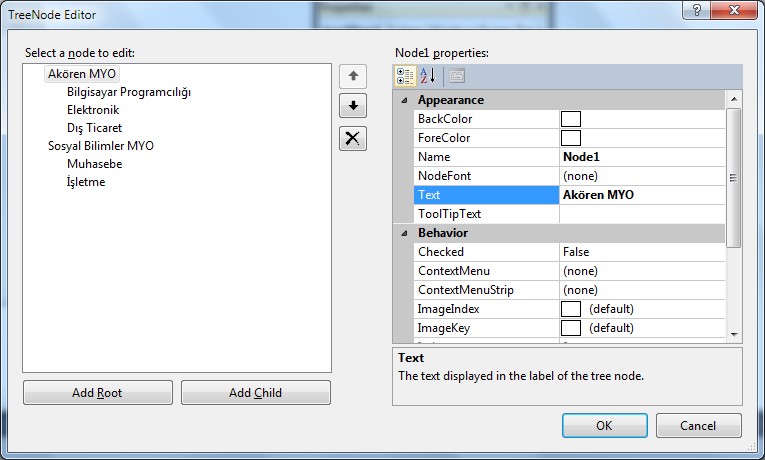
Resimlerin bu özellikte belirtilen renkteki kısımları saydam olur.

# TreeView Kontrolü:

Öğeleri ağaç yapısında gösteren bir kontroldür. Bu kontrol, içinde bulunan öğeleri hiyerarşik bir yapıda görüntüler. Eklenen her öğe bir düğümü (*node*) temsil eder ve düğümler birleşerek ağaç yapısını oluştururlar. Her düğüme alt düğüm eklenebilir.



***TreeView Özellikleri :***



***Nodes :***

Kontrole eklenecek düğümler ve alt düğümler bu özellik ile belirlenir.

*Nodes*

özelliğinde iken

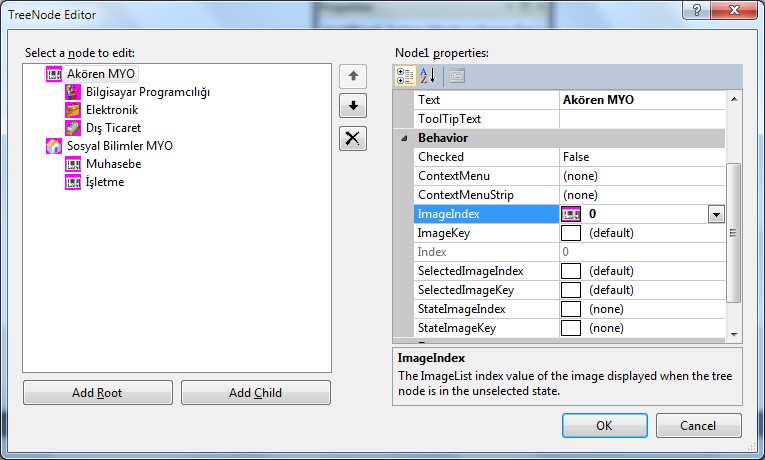
düğmesine tıklandığında aşağıdaki pencere açılır.

***ImageList :***

Düğümlerde gösterilecek resimlerin seçileceği liste şeklinde resimleri tutan kontrol ismidir.

***ImageIndex :***

Bu özelliğin kullanılabilmesi için, kontrolün *ImageList* özelliğinin belirlenmiş olması gerekir. Resmin *ImageList* kontrolündeki indisini belirtir. Yani *ImageList* kontrolündeki birinci resim seçilecekse *ImageIndex* değeri 0 dır.



***TreeView Olayları :***

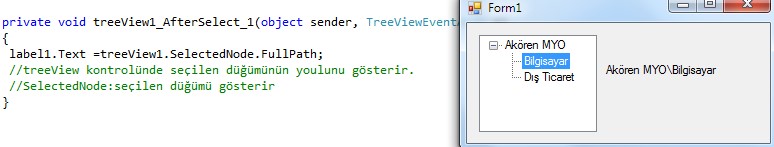
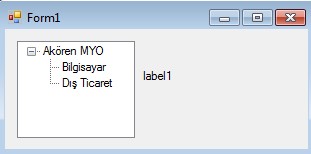
***BeforeSelect :***

Düğüm seçilmeden önce gerçekleşir

***AfterSelect :***

Düğüm seçildikten sonra gerçekleşir

## Örnek Uygulama:



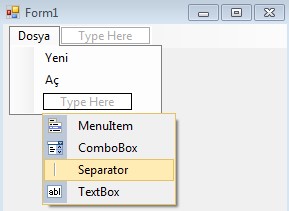
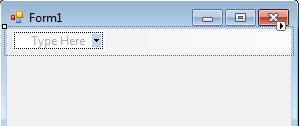
Seçilen

düğümün yolunu gösteren uygulama.

# menuStrip Kontrolü:

Formlarda menü oluşturmak için kullanılan kontroldür.

Form üzerine *menuStrip* kontrolü yerleştirdikten sonra, aşağıda gösterildiği gibi *“Type Here”* yazan kısma menü



isimleri yazılır.

Menü

öğelerinde

&(ampersand)

işareti

eklenen

menü

karakterine

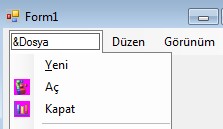
ALT+karakter

ile

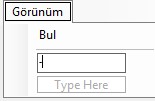
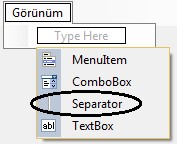
ulaşılabilir.

Aşağıdaki

formda “Dosya” menüsüne ALT+D ile ulaşılabilir.



Menü öğeleri arasına çizgi koymak için “*Separator”* seçeneği kullanılır ve iki şekilde yapılabilir. *“Type Here”*



yazan kısımda

–

(

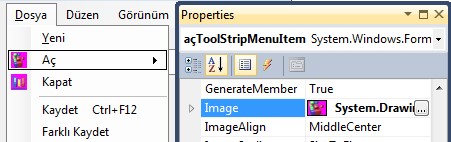
tire) işareti yazılır yada “

*Separator*

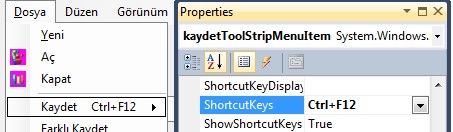
*”*

seçilir.

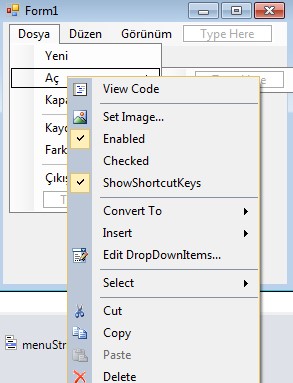
Menü seçeneklerinde Resim göstermek için *“Image…”* seçilir.



*“ShowShortcutKeys”* ile menü öğelerinde kısayol aktif olur. Aktif olan kısayolun gösterilmesi için *ShowShortcutKeys* özelliğinin *True* olması gerekir.



*“Convert To”* ile menü seçeneği kontrolü diğer kontrollere dönüştürülebilir. *“Insert”* ile araya yeni menü seçeneği eklenebilir. *“EditDropDownItems…”* ile menü seçenekleri yeni açılan bir pencere ile düzenlenebilir.



Menü öğelerinin diğer özelliklerini değiştirmek için öğe seçildikten sonra *“Properties”* penceresinden yapılır.

*Checked :* true olursa menü öğesinin yanında seçili olduğunu gösteren işaret gösterilir.

*CheckOnClick :* menü öğesi tıklanarak seçili olup olmaması değiştirilir. *DisplayStyle :* menü öğesinde sadece yazı ve resim yada her ikisinin birlikte gösterilmesini sağlar.

*ShortcutKeyDisplayString :* menü öğesinde kısayol olarak gösterilecek karakter yazılır.

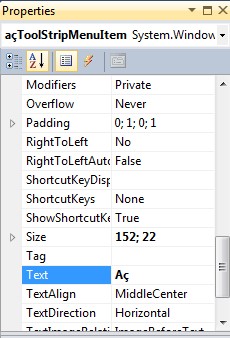
*ShortcutKeys :* kısayol olarak seçilecek tuşlar belirlenir.

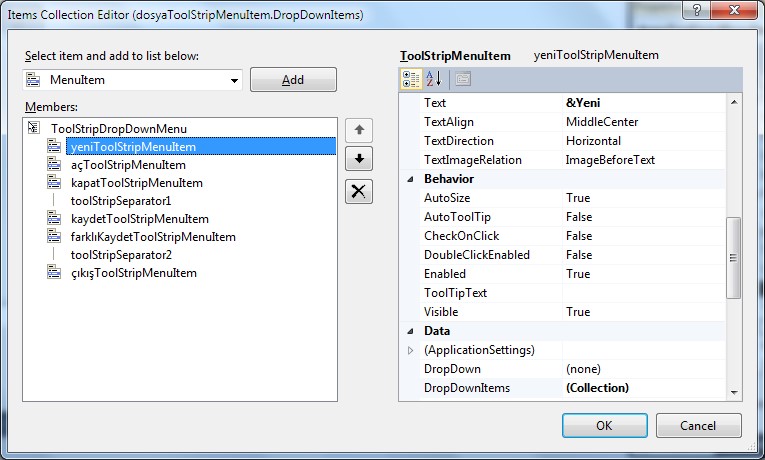
*ShowShortcutKeys :* kısayolun menü öğesi yanında gösterilip gösterilmemesini belirler.

*Text :* menü öğesinde gösterilecek yazı.

*TextAlign :* gösterilen yazının konumunu belirler.

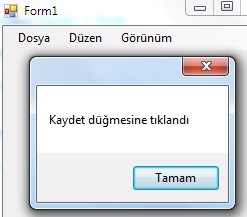
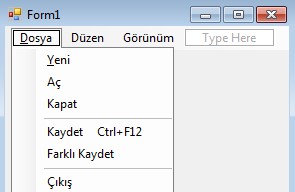
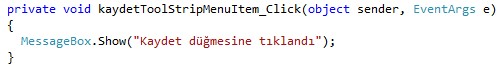
*TextDirection :* gösterilen yazının dikey yada yatay olmasını belirler. TextImageRelation : gösterilen yazı ve resmin birbirlerine göre nerede gösterileceğini belirler.

İkinci yol olarak *menuStrip* kontrolü seçildikten sonra özellikler penceresinden *Items* özelliği seçilerek yada menü öğeleri seçildikten sonra *DropDownItems* özelliği seçilerek açılan pencereden de menüler oluşturabilir.



***Çalışma Zamanında Menüleri Değiştirmek :***

Menüler çalışma zamanında dinamik bir şekilde yönetilebilir. Örneğin, menü pasif duruma getirilebilir, gizlenebilir yada çalışma zamanında bir menü öğesi eklenebilir veya başka bir menü öğesiyle birleştirilebilir.



***Örnek Uygulama:***

Gösterildiği gibi bir menü oluşturunuz ve kodları

“kaydet”

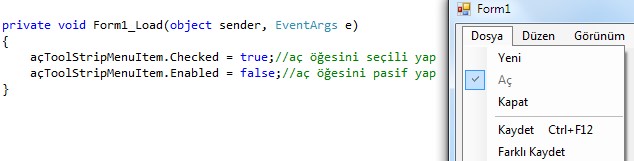
Öğesinin

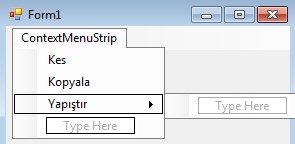
*Click*

olayına

yazınız.

## Örnek Uygulama:





**contextMenuStrip Kontrolü**

**:**

İçerik

menüleri

yada

popup

menüler

olarak

bir

kontrol

üzerinde

sağ

tıklama

ile

açılan

menüler

oluşturmak

için

kullanılır.

*contextMenuStrip*

öğelerini

kontrolünde

menü

oluşturmak

*menuStrip*

kontrolünde

olduğu

gibidir.

Hangi

kontrolde

içerik

menüsü

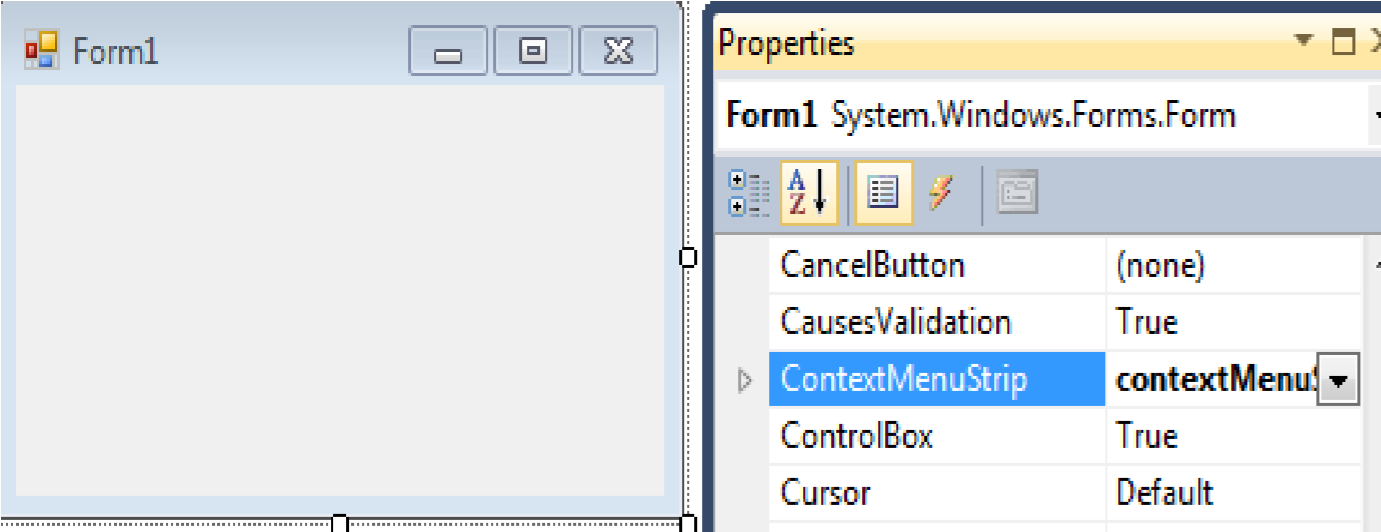
gösterilecekse,

o

kontrolün

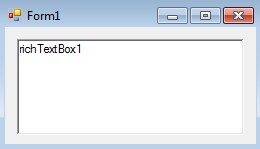
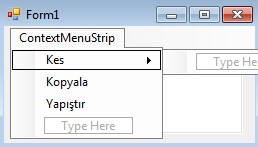
*contextMenuStrip*

özelliği seçilmelidir.



## Örnek Uygulama:

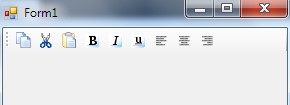
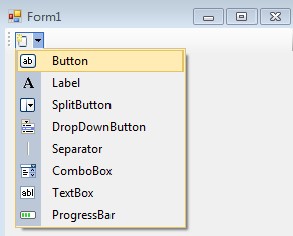
Gösterildiği gibi form üzerine *richTextBox1* ve *ContextMenuStrip1* kontrollerini yerleştiriniz. *richTextBox1* kontrolünün “*ContextMenuStrip”* özelliğini *ContextMenuStrip1* seçiniz. *ContextMenuStrip1* kontrolünde yazılacak menü öğeleri şekilde gösterilmişti. Menü öğelerinin *Click* olayına yazılacak kodlar gösterilmiştir.





# toolStrip Kontrolü:

Formun üst tarafında araç çubuğunda kısayollar bulunan kontroldür. Araç çubuğunda gösterilebilecek

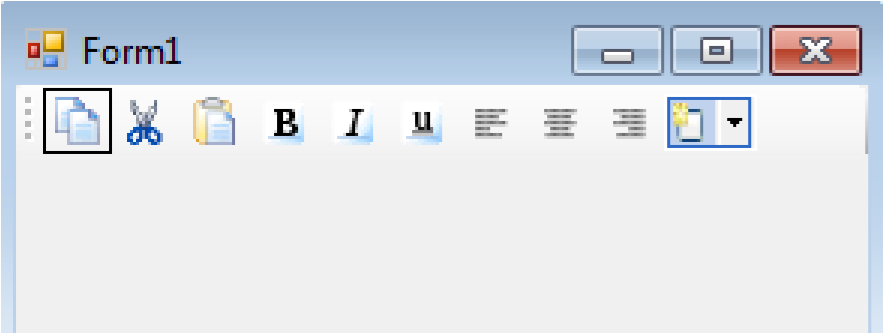


kontroller aşağıdaki resimde gösterilmiştir.

*Button*

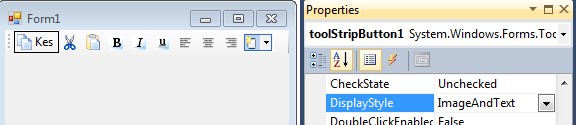
seçilerek kısayol öğeleri oluşturulur.

Araç çubuğunda oluşturulan düğme kontrolleri seçilerek birçok özelliği *Properties* penceresinden değiştirilebilir.



***DisplayStyle :***

Düğmede *text*, resim yada ikisinin birden gösterilmesi ayarlanır varsayılan değeri *Image* dir.



***Text :***

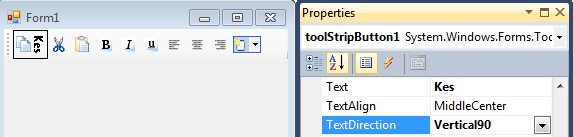
Düğmede gösterilmesi istenen yazı girilir.

***TextAlign :***

Düğmede gösterilecek yazının düğmedeki hizasını ayarlar.

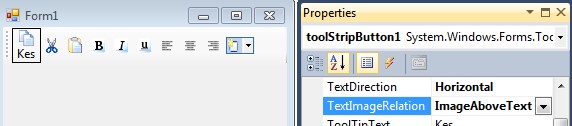
***TextDirection :***

Düğmede gösterilecek yazının yönünü ayarlar.

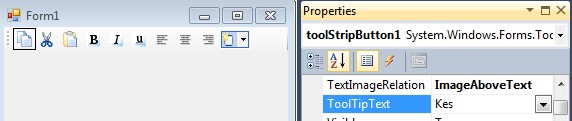


***TextImageRelation :***

Text ile resmin birlikte nasıl gösterileceğini belirler.



***ToolTipText :***



Araç çubuğundaki kısayol düğmeleri hakkında açıklama göstermek için kullanılır.

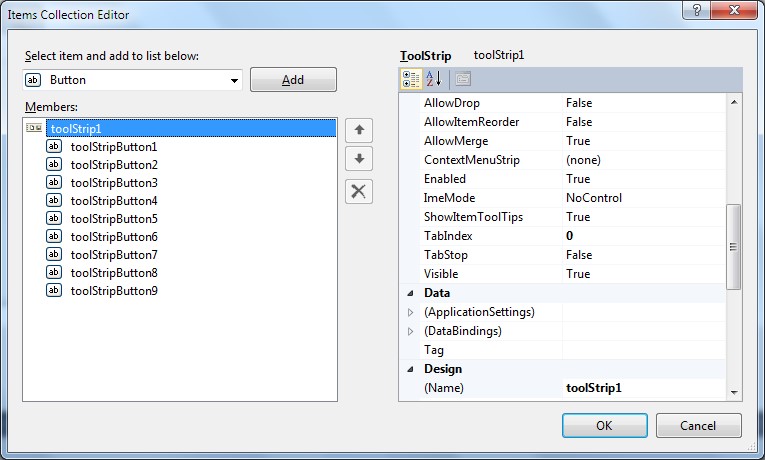
***toolStrip Özellikleri :***

***GripStyle :***

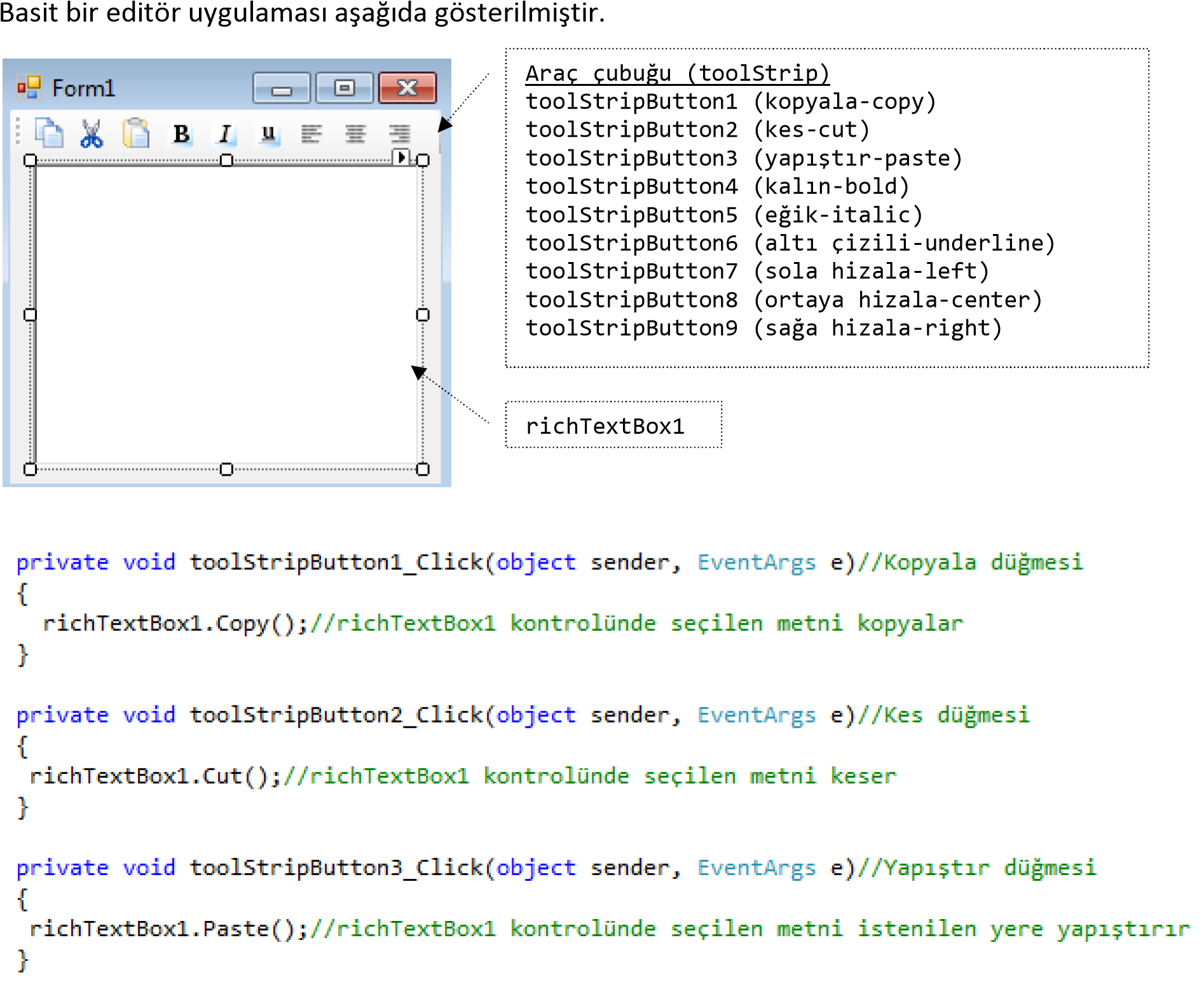
Araç çubuğunun en solunda dikey noktaların gösterilmesini sağlar. 

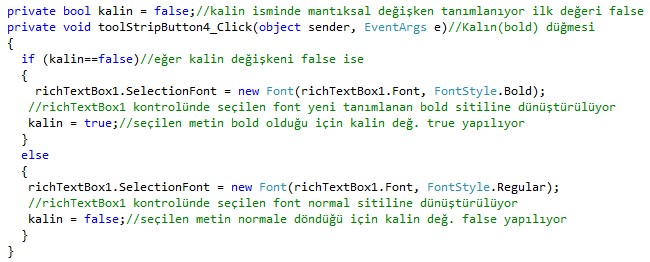
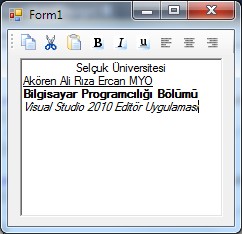
***Items :***

Araç çubuğunun (toolStrip) ve üzerindeki düğmeler bu özellik ile açılan pencerede yapılabilir.



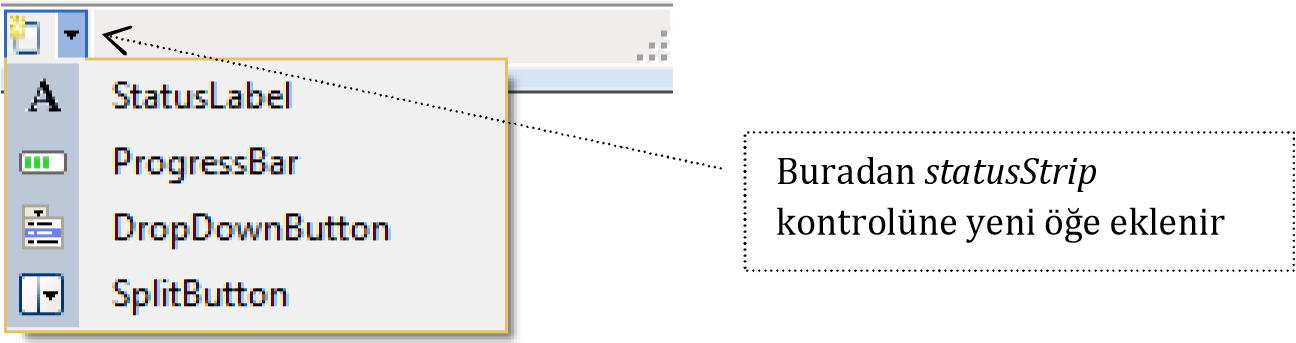
## Örnek Uygulama:





# statusStrip Kontrolü:

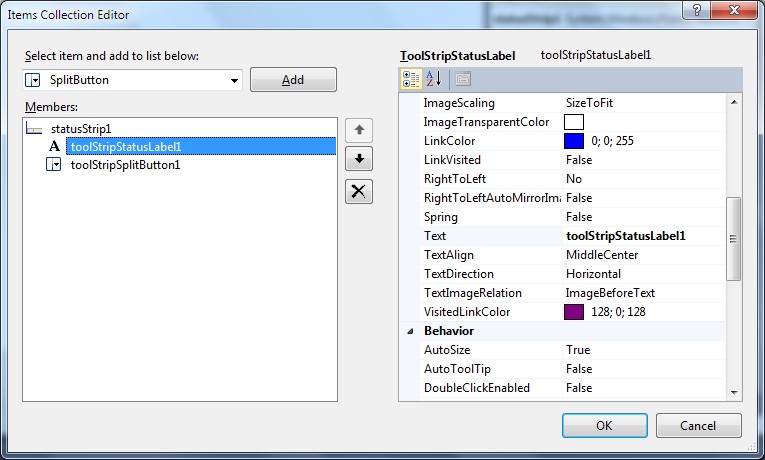
Formların en altında kullanıcıya bilgi veren durum çubuğudur. *StatusLabel*, *ProgressBar*, *DropDownButton* ve *SplitButton* gibi öğelerden birisi seçilerek *statusStrip* kontrolüne eklenir.



***statusStrip Özellikleri :***

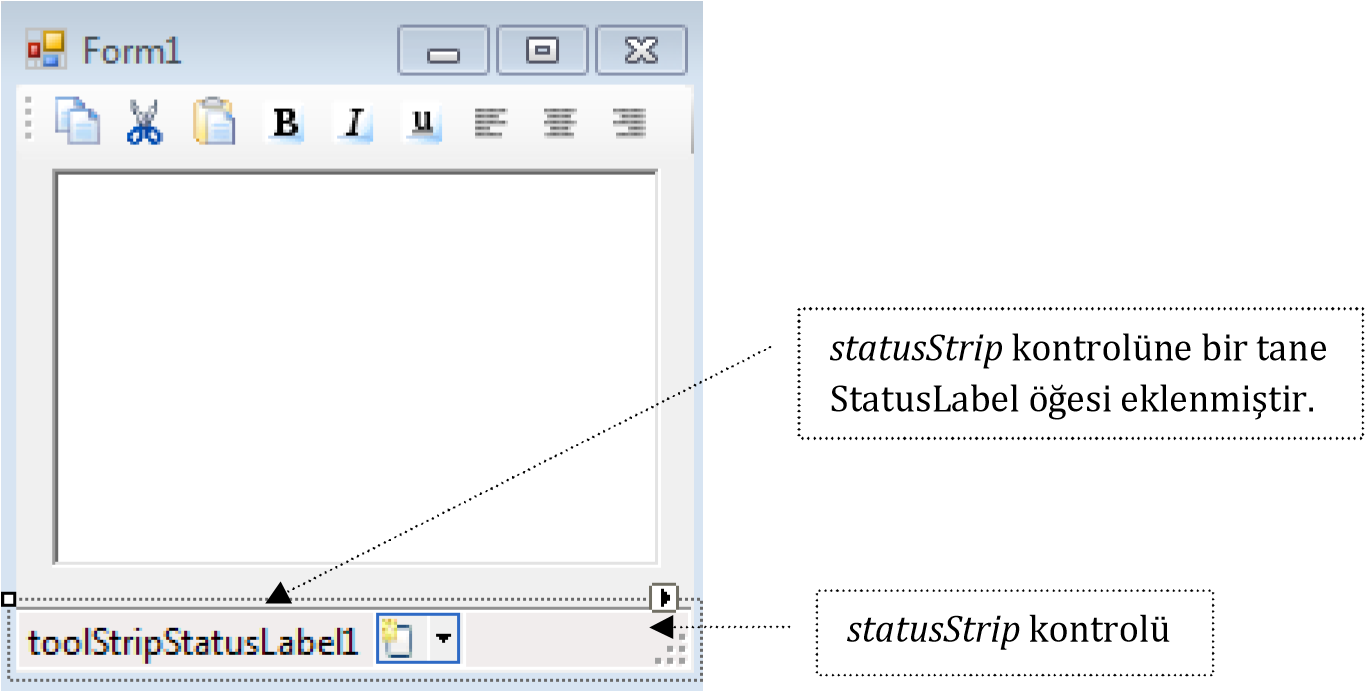
***Items :***

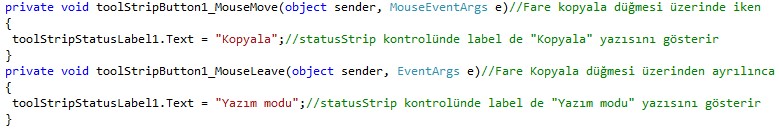
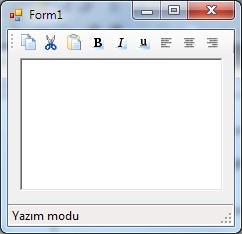
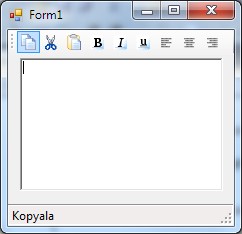
*statusStrip* kontrolüne yeni öğe eklemek, düzenlemek yada silmek için yeni bir pencere açar ve buradan öğelerin özelliklerinin ayarlanmasını sağlar.



## Örnek Uygulama:

Yukarıda toolStrip kontrolünde yapılan editör uygulamasına *statusStrip* kontrolü ekleyelim.





# trackBar Kontrolü:

*trackBar* kontrolü *ScrollBar* kontrollerine (kaydırma çubukları) benzer yapıdadır ve bir değer artışı yada azalışı görevini yapar. Klavyede bulunan yukarı, aşağı, sağ, sol okları ve PageUp, PageDown tuşları ile *trackBar* kontrolünün değeri değiştirilebilir.

***trackBar Özellikleri :***

***LargeChange :*** *trackBar* kontrolünde fare ile tıklamada kaydırma çubuğunun hareketinin en büyük değeridir. Varsayılan değeri 5 dir ve her fare ile tıklamada 5 adım hareket eder.

***Maximum :*** *trackBar* kontrolünün alabileceği en büyük değerdir.

***Minumum :*** *trackBar* kontrolünün alabileceği en küçük değerdir.

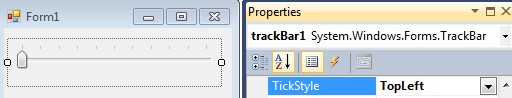
***Orientation :***

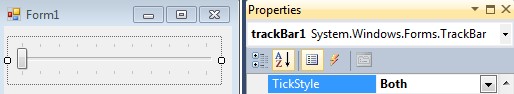
*trackBar* kontrolünün formda, yatay (horizontal) yada dikey (vertical) olmasını belirler.

***TickFrequency :*** *trackBar* kontrolündeki çizgi sayılarının sıklığını belirler. Örneğin; *trackBar* kontrolünün en küçük değeri 0 ve en büyük değeri 10 ise ve *TickFrequency* değeri 1 ise *trackBar* kontrolünde 10 tane çizgi gösterilir.

***TickStyle :***

Çizgilerin kontrolde nerede gösterileceğini belirler.





***Value :***

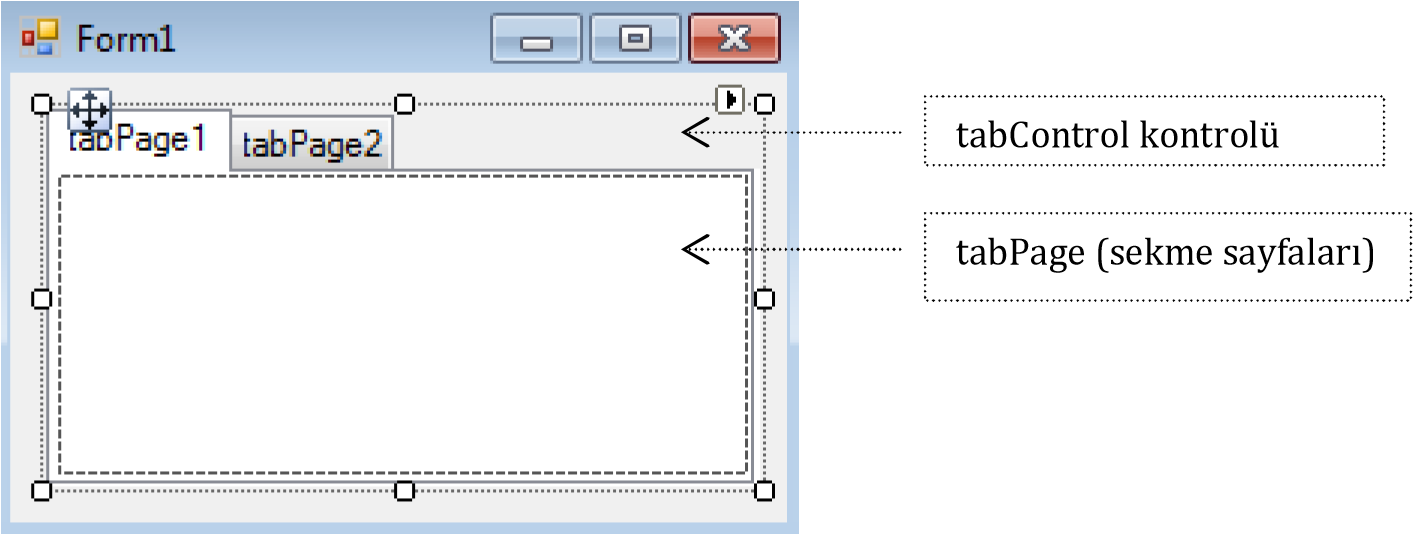
*trackBar* kontrolündeki göstergenin sayısal değerini gösterir.

## Örnek Uygulama:



# tabControl Kontrolü:

Sekme şeklinde sayfalar içeren bir kontroldür ve her sekme sayfasının kendine ait özellikleri bulunmaktadır.



***trackBar Özellikleri :***

***SelectedTab :***

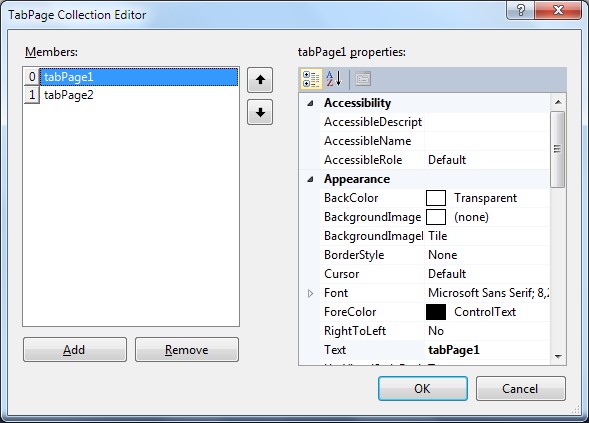
Aktif olan sekme sayfasını belirler.

***SelectedIndex :***

Aktif olan sekme sayfasının indisini belirler.

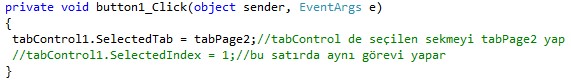
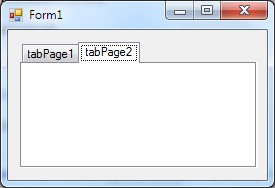
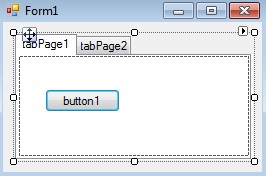
***TabPages :***

Sekme sayfalarının eklendiği, düzenlendiği ve silindiği yeni bir pencere açar ve sekme sayfaları buradan ayarlanır.



## Örnek Uygulama:

Aşağıdaki formu tasarlayarak form üzerinde *tabControl1* oluşturarak *tabPage1* sayfasına bir düğme



yerleştiriniz. Bu düğme tıkland

ığında

*tabPage2*

açılacaktır.

# pictureBox Kontrolü:

Form üzerinde resim gösteren kontroldür.

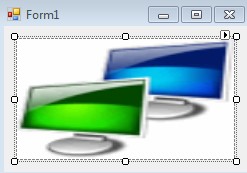
***pictureBox Özellikleri :***

***Image :***

*pictureBox* kontrolünde gösterilecek resmin kaynağını belirler.

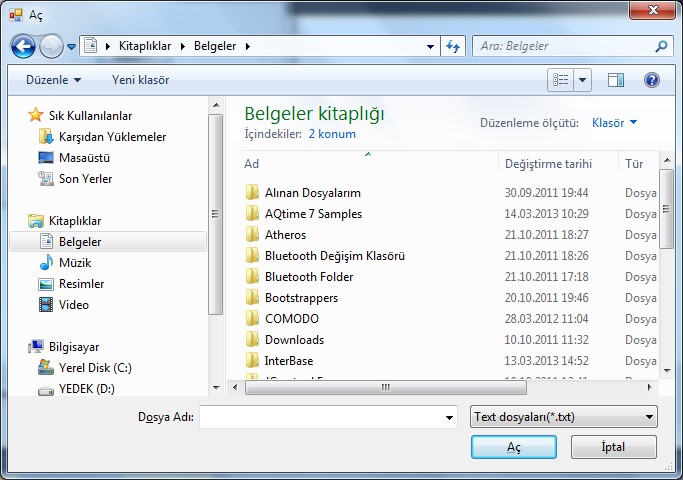
***SizeMode :***

Resmin kontrolde nasıl gösterileceğini belirler.



# openFileDialog Kontrolü:

Dosya açma diyalog penceresini açan kontroldür ve bu sayede istenilen dosya seçilir. *openFileDialog.ShowDialog()* komut ifadesi ile aşağıdaki pencere açılır.



***openFileDialog Özellikleri :***

***FileName :***

*openFileDialog* dosya açma penceresinde seçilen dosyanın adını belirler.

***Filter :***

*openFileDialog* dosya açma penceresinde istenilen uzantıdaki dosyaların listelenmesini ayarlar. Listelenmesi istenilen dosya uzantıları aşağıda gösterildiği şekilde yazılmalıdır.

Text dosyaları(\*.txt)|\*.txt|Tüm dosyalar(\*.\*)|\*.\*

Filtre uygulanacak dosya uzantıları arasına kesme (|) karakteri konulmalıdır.

Örnek kod: openFileDialog1.Filter = " (\*.jpg)|\*.jpg|(\*.png)|\*.png";

***FilterIndex :***

Filtre yapılırken varsayılan dosya uzantısını belirler. *Filter* kısmında yazılan dosya uzantılarından hangisi varsayılan olarak gösterilecekse onun indis numarası yazılır. Örneğin; ***Text dosyaları(\*.txt)|\*.txt|Tüm dosyalar(\*.\*)|\*.\****

Yazımında varsayılan olarak, ilk yazılan \*.txt uzantılı dosyalar gösterilir ancak \*.\* uzantılı dosyaların varsayılan olarak gösterilmesi için *FilterIndex* özelliğine 2 yazılır.

Örnek kod: openFileDialog1.Filter = " (\*.jpg)|\*.jpg|(\*.png)|\*.png";

openFileDialog1.FilterIndex = 1; // varsayılan olarak jpg uzantıları göster

***InitialDirectory :***

*openFileDialog* penceresinde varsayılan olarak açılması istenilen klasörün yolunu belirler.

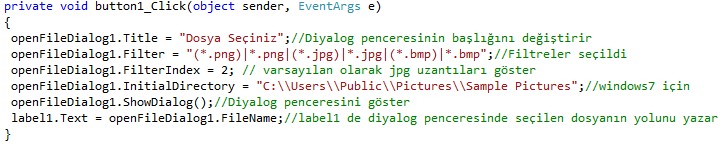
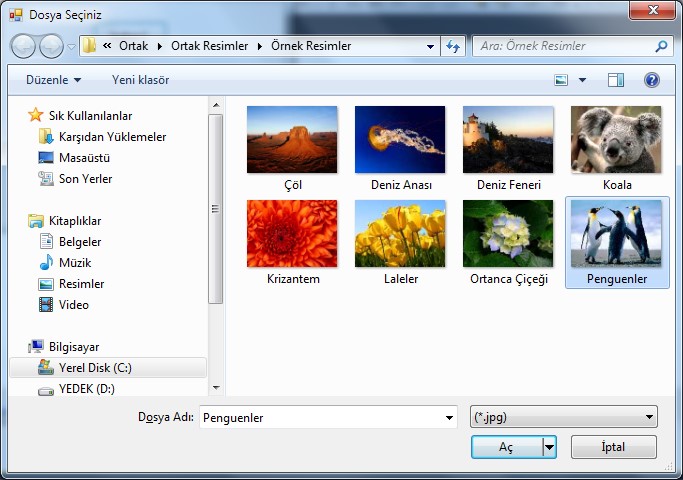
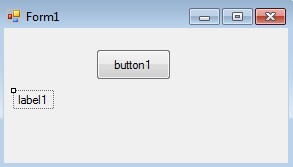
Örnek kod: openFileDialog1.InitialDirectory="C:\”

***Title :***

*openFileDialog* penceresinin başlığını belirler ancak, *openFileDialog1.ShowDialog()* komutundan önce kullanılmalıdır. Çünkü pencere açıldıktan sonra Title' a değer atansa bile formun başlığı değişmeyecektir.

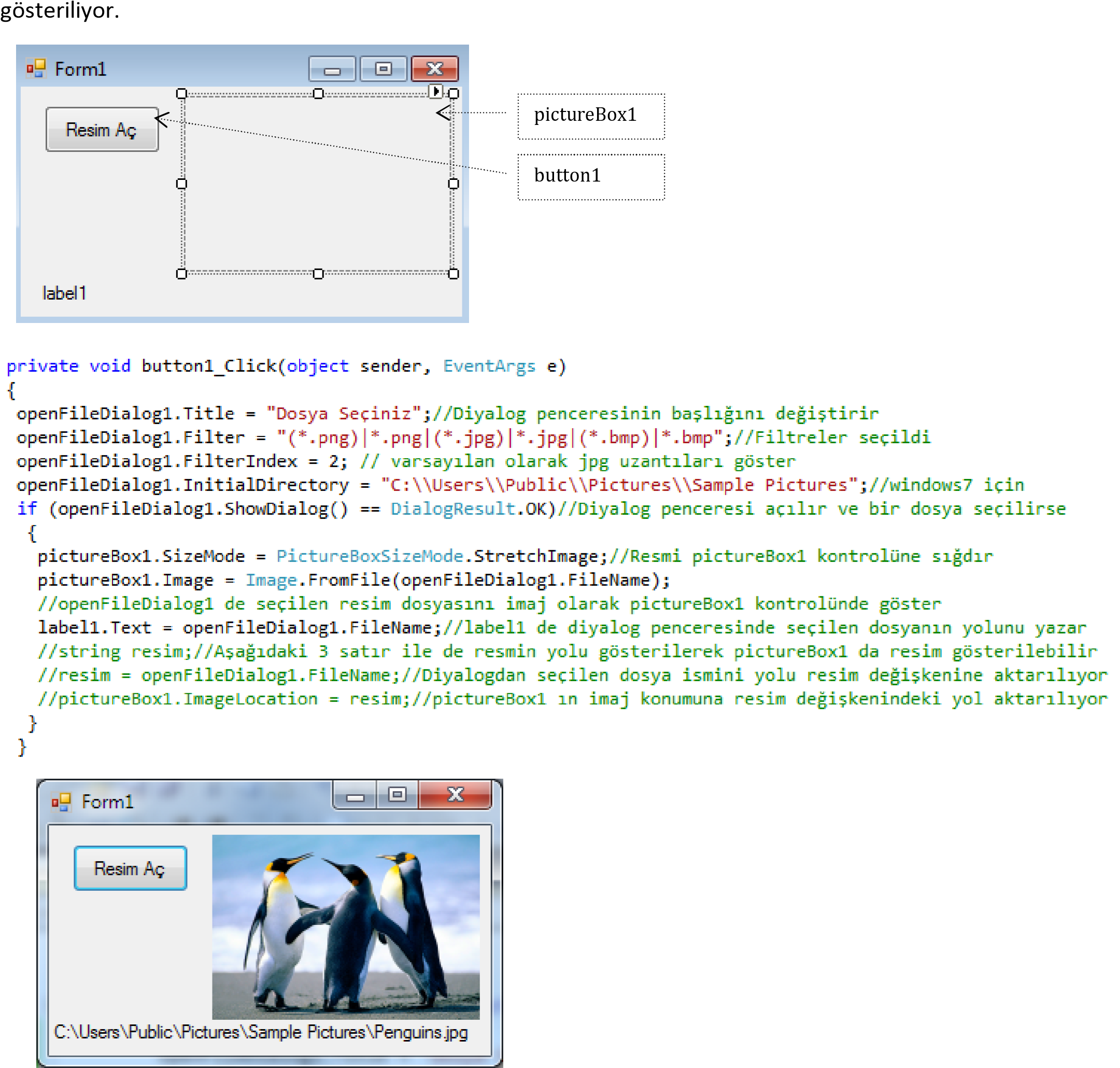
## Örnek Uygulama:

Aşağıdaki formu tasarlayınız. Burada *openFileDialog* penceresinin özellikleri kod ile değiştirilmiştir. Bu uygulamada düğmeye tıklandığında *openFileDialog* penceresi açılacaktır ve seçilen dosyanın yolu *label1* de yazdırılacaktır.



## Örnek Uygulama:

Bu uygulamada seçilen bir resim dosyasının yolu label1 de gösteriliyor ve seçilen resimde pictureBox1 de

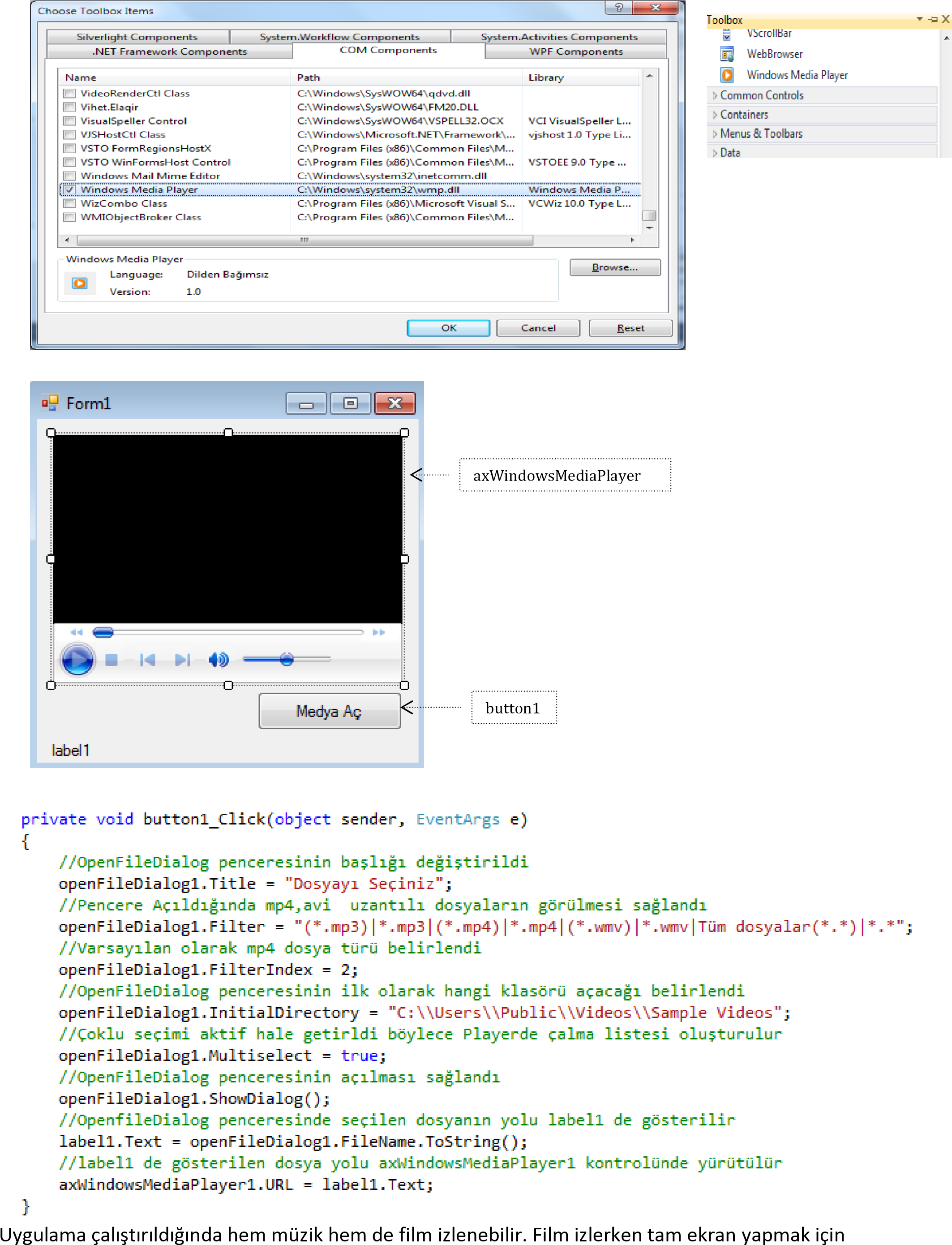


## Örnek Uygulama:

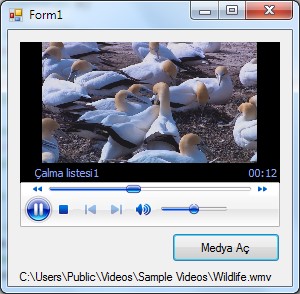
Seçilen müzik, video gibi medyaları gösteren bir uygulama.

Öncelikle forma bir adet *WindowsMediaPlayer* kontrolü koyunuz. Eğer *toolbox* da *WindowsMediaPlayer* kontrolü yoksa aşağıdaki adımları takip ediniz.

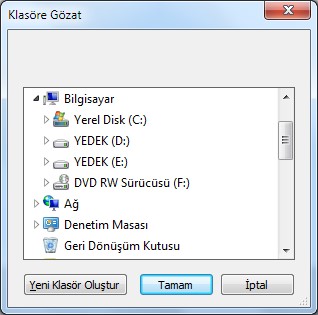
1. *toolbox* üzerinde herhangi bir nesneye sağ tıklanır ve *“Choose Items”* seçeneği seçilir
2. Açılan pencerede *“Com Components”* sekmesinden *WindowsMediaPlayer* işaretlenir ve Ok tıklanır.
3. *toolbox* da bu işlemlerden sonra *WindowsMediaPlayer* kontrolü oluşur.



*axWindowsMediaPlayer1* kontrolünün *fullScreen* özelliğinin *True* yapılması gerekir.

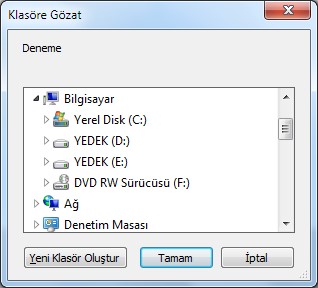


**folderBrowserDialog Kontrolü:** *folderBrowserDialog* kontrolü, bilgisayardaki klasörlerin dizin şeklinde gösterilmesini sağlar.



***folderBrowserDialog Özellikleri :***

***Description :***



Bu özellik

,

f

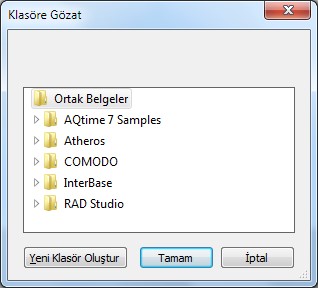
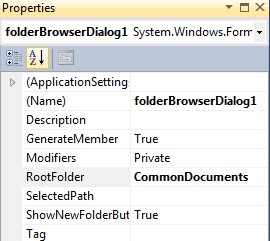
*olderBrowserDialog*

penceresinin

üst kısmında açıklama

görüntülenmesini sağlar.

***RootFolder :***



*folderBrowserDialog*

penceresinde

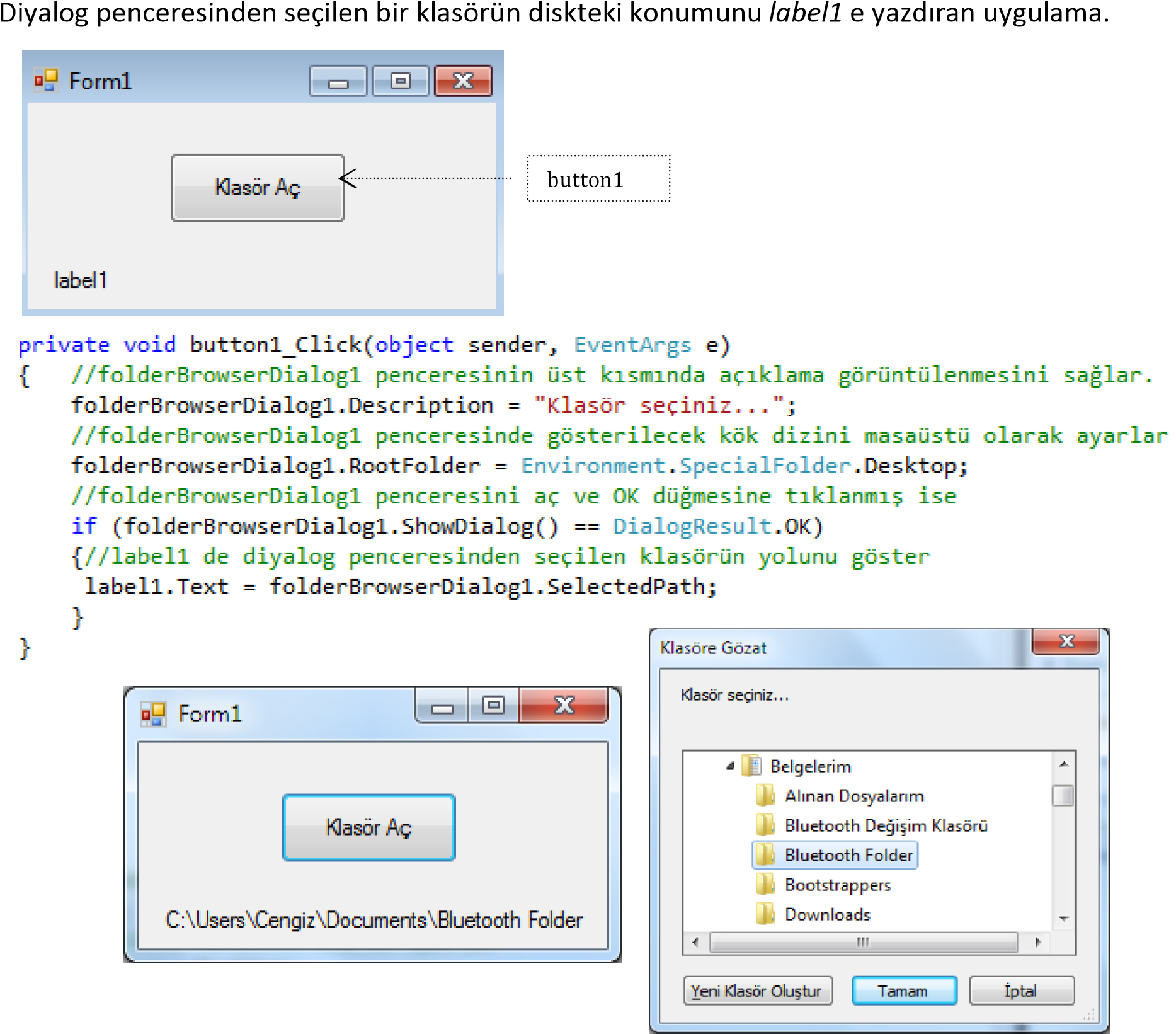
gösterilecek klasörlerin kök dizinini belirler.

***SelectedPath :***

*folderBrowserDialog* penceresi ilk açıldığında hangi dizinin varsayılan olarak açılacağı bu özellik ile belirlenir.

***ShowNewFolderButton :*** *folderBrowserDialog* penceresinde “Yeni Klasör Oluştur” düğmesinin görünür olup olmayacağının ayarlandığı özelliktir.

## Örnek Uygulama:



Örnekler

## C# TİMER'I DİNAMİK OLARAK KONTROL ETMEK

Yazar: [furkanuyanik](http://www.yazilimindunyasi.com/user/furkanuyanik/)Kategori: [C# Form Application Örnekleri](http://www.yazilimindunyasi.com/c-windows-form-application/)Tarih: [19-11-2015, 23:40](http://www.yazilimindunyasi.com/2015/11/19/)

1

  
  
Merhaba arkadaşlar, C# üzerinden yazdığım bu programın amacı aşağıdaki gibidir:  
1'den 30'a kadar sayan bir timer bulunmakta. Timer başlangıçta saymaya başladığında eğer süre dolana kadar "Saydır" butonuna basarsanız 1 artacak ve zamanı dolduğunda sıfırdan tekrar başlayacak. Butona ne kadar basarsanız o kadar tekrar ederek 0dan 30a kadar sayaç o kadar sayacak. Temel amacımız bir timer arkadan çalışırken dinamik olarak kontrol etmektir.  
  
İyi çalışmalar!

 private void btn\_Arttir\_Click(object sender, EventArgs e)

        { if (islemkod!=1)

            { if (lbl\_Saydir.Text == "0")

                timer1.Start();

            lbl\_Saydir.Text=(Convert.ToInt32(lbl\_Saydir.Text)+1).ToString();

            }

        else

            {   lbl\_Saydir.Text = "1";

                btn\_Arttir.Text = "Arttır";

                timer1.Start();

                islemkod = 0;

                sayac = 0;

                sayma\_sayisi = 0;

            }

        }

        int sayac = 0,sayma\_sayisi=0,islemkod=0;

        private void timer1\_Tick(object sender, EventArgs e)

        {  sayac++;

            lbl\_Zaman.Text = sayac.ToString();

            if (sayac==30)

            {  sayma\_sayisi++;

                if (sayma\_sayisi == Convert.ToInt32(lbl\_Saydir.Text))

                {  timer1.Stop();

                    MessageBox.Show("İşlem bitmiştir.");

                    btn\_Arttir.Text = "Yeniden";

                    islemkod = 1;  }

                else

                {

                    sayac = 0;

                }

            }

        }

