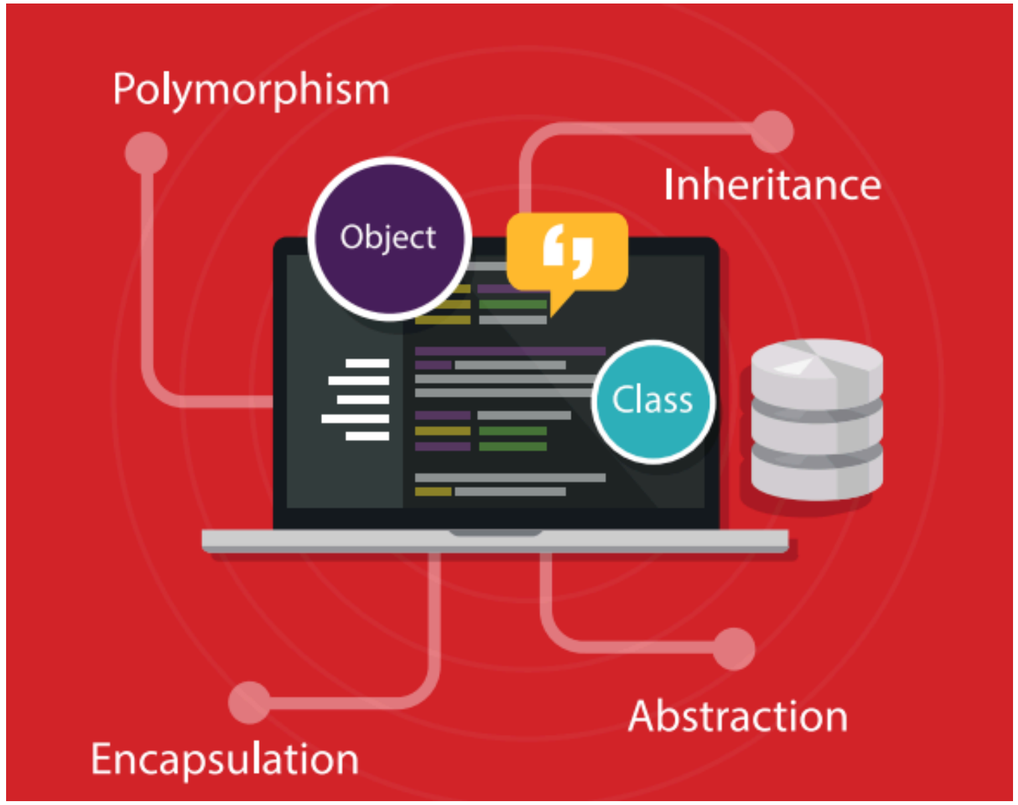
NESNE TABANLI PROGRAMLAMA-13

SINIFLAR (CLASS)



KONULAR

3.1. SINIFLAR VE NESNELER
3.2. SINIF ÖZELLİKLERİ
3.3. METOT OLUŞTURMA VE ÇAĞIRMA
3.4. METOTLARI AŞIRI YÜKLEME
3.5. ERİŞİM BELİRLEYİCİLER
3.6. KAPSÜLLEME, KALITIM VE ÇOK BİÇİMLİLİK

ANAHTAR KELİMELER

Sınıf, nesne, alan, özellik, kapsülleme, metot, kalıtım, soyut sınıf, arayüz, çok biçimlilik, statik sınıf, isimsiz sınıf, mühürlü sınıf, parçalı sınıf, enums

NELER ÖĞRENECEKSİNİZ?

- Nesne tabanlı programlama mantığı
- Nesne tabanlı programlamanın temel prensipleri
- Sınıf ve nesneler oluşturma
- Sınıflarda alan, özellik ve metot ögeleri
- Erişim belirleyicileri
- Metotlar
- Değer ve referans kavramları
- Sınıflarda kalıtım özellikleri
- Soyut sınıf, arayüz ve çok biçimlilik kavramları
- Statik, isimsiz, mühürlü ve parçalı sınıflar
- Numaralandırma mantığı

1.NESNE TABANLI PROG. GİRİŞ

Nesne Tabanlı Programlama (OOP), programlamayı daha anlaşılır, düzenli ve yeniden kullanılabilir hale getiren bir tekniktir. Nesneler (objects) ve sınıflar (classes) kullanarak yazılım geliştirmenin temelini oluşturur.

Diyelim ki büyük bir apartmanda yaşıyoruz. Apartmanda birçok daire (nesne) var ve bu dairelerin her birinin bir kapı numarası, büyüklüğü, odaları gibi özellikleri bulunuyor.

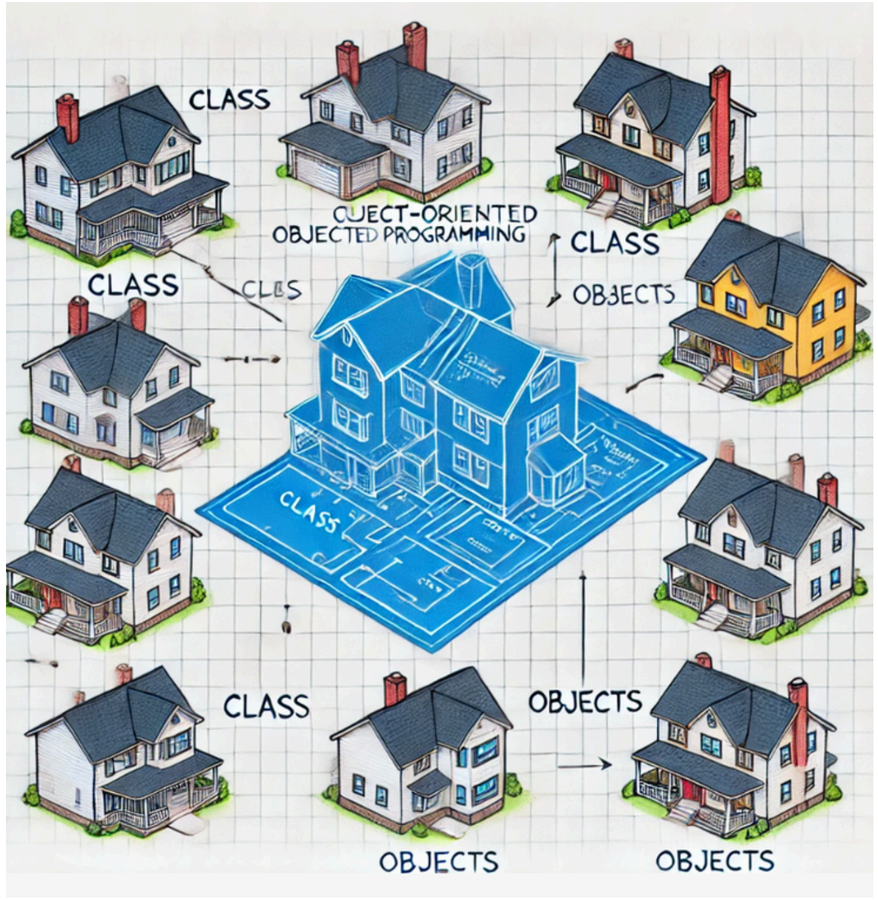
💡 **OOP Mantığı:**

- **Sınıf (Class) → Plan / Şablon**
- **→ Bir apartmanın genel tasarımıdır (Tüm daireler için geçerli olan kurallar ve özellikler).**
- **Nesne (Object) → Gerçek Dünya Örneği**
- **→ Apartmandaki her bir daire, bu planın gerçek bir örneğidir.**

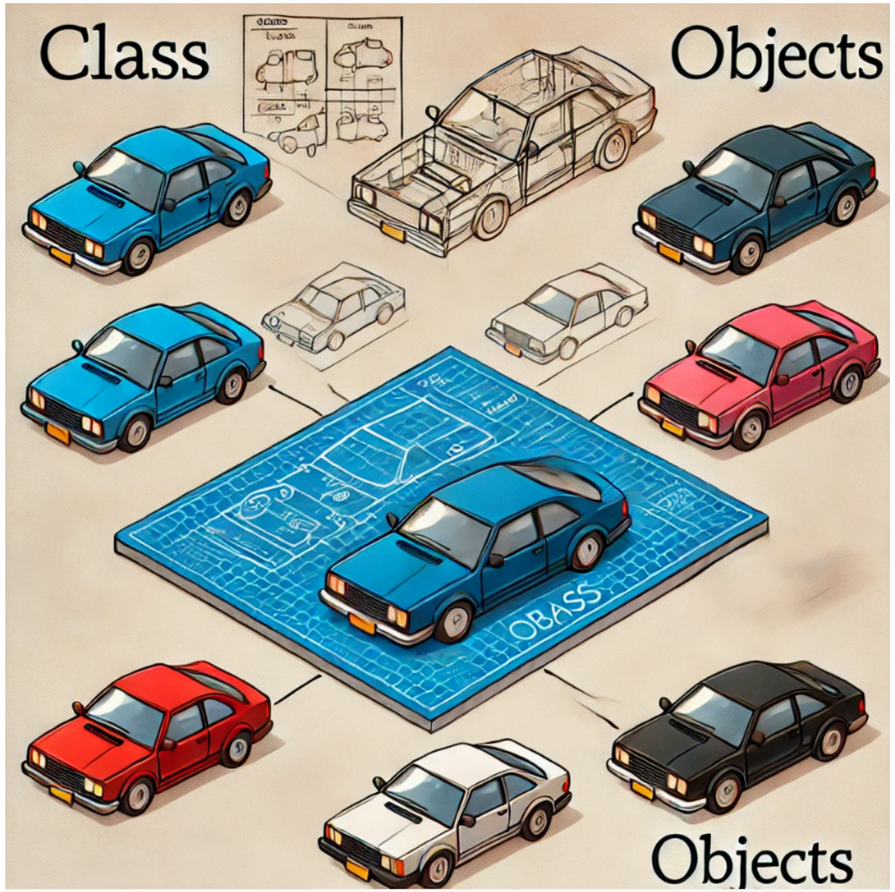
Örnek:

- **Sınıf: "Araba" (Marka, model, hız gibi özellikleri olan bir şablon)**
- **Nesneler: "BMW", "Toyota" gibi gerçek arabalar**

1.NESNE TABANLI PROG. GİRİŞ



bir **mimari plan (sınıf**) ile bu plana göre inşa edilmiş **farklı evler (nesneleri)**

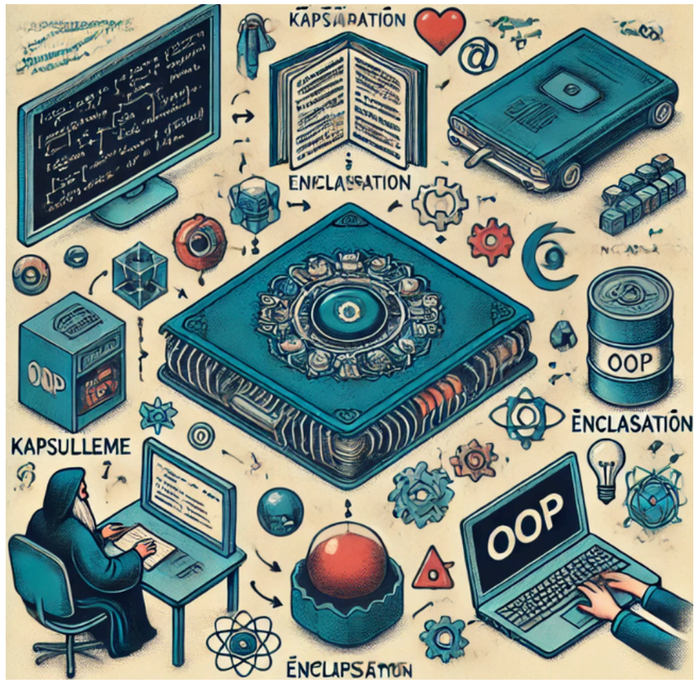


bir **araba tasarım planı (sınıf - class)** ve bu plana göre üretilmiş **farklı araba modelleri (nesneler - objects**)

1.1. N.T.P. PRENSİPLERİ

a) Soyutlama (Abstraction): Karmaşıklığın azaltılması anlamına gelir. Örneğin otomobillerde gaz pedalına basıldığında otomobil hızlanır ancak arka planda olan bitenler çoğu kişi için önemsizdir.

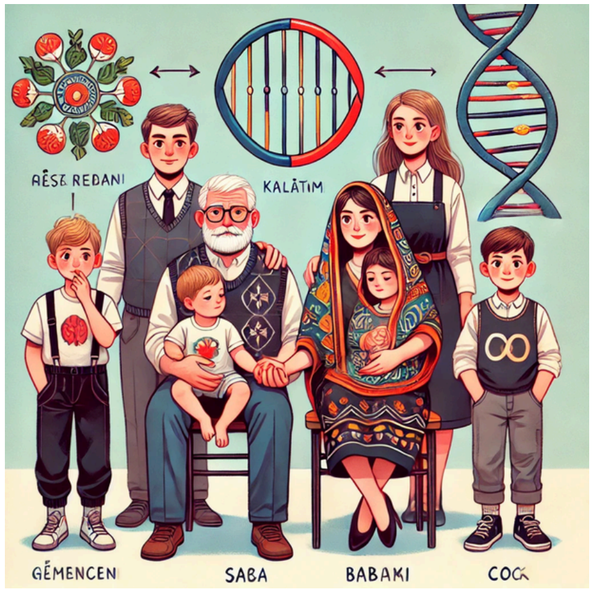
****** Sınıflar - Nesneler Oluşturma**



b) Sarmalama veya Kapsülleme (Encapsulation): Sadece istenilen bilgilerin dış dünyaya açılması, hassas veya özel bilgilerin gizlenmesi anlamına gelir. "Banka hesabına para yatır." komutu verildikten sonra T.C. Kimlik Numarası ve şifre bilgilerinin gizlenmesi buna örnek verilebilir.

- Kapsülleme, verilerin korunmasını sağlar.
- Dışarıdan erişimi kontrol etmek için private ve public gibi erişim belirleyiciler kullanılır.

1.1. N.T.P. PRENSİPLERİ



c) Kalıtım (Inheritance): Var olan özelliklerin aktarılması anlamına gelir. Örneğin bütün kedi türlerinin dört ayaklı olması ortak bir özelliktir. Bir Van kedisi, kedilerin tüm özelliklerini taşırken ayrıca kendine has özellikleri de barındırır.

- Kalıtım, bir sınıfın özelliklerini başka bir sınıfa aktarmasını sağlar.
- Ana sınıf (Parent Class) → Alt sınıf (Child Class) türetebilir.

ç) Çok biçimlilik (Polymorphism): Farklı tiplere ait olan ortak özellikleri tanımlama işlemidir. Örneğin farklı hayvan türleri farklı sesler çıkarır zira "ses çıkarma" eylemi ortak bir özelliktir.
Aynı metodu, farklı sınıflarda farklı şekilde kullanabilme yeteneğidir.

1.1. N.T.P. PRENSİPLERİ

Özetlersek....

Kavram	Anlamı	Örnek
Sınıf (Class)	Şablon, plan	Araba sınıfı
Nesne (Object)	Gerçek dünya örneği	Toyota, BMW
Kapsülleme (Encapsulation)	Verileri saklama	Öğrenci bilgilerini `private` yapma
Kalıtım (Inheritance)	Özellikleri miras alma	Hayvan → Kedi
Polimorfizm (Polymorphism)	Metotları değiştirme	Kedi: Miyav!, Köpek: Hav!

ETKİNLİK - 1

"Bir oyun karakteri tasarlayın!" diyerek kendi sınıfımızı oluşturmak istesek adım adım nasıl olurdu ?

OOP Prensibi	Tanımı	Oyun Karakteri İçin Örneği	Gerçek Hayattan Benzetme
Kapsülleme (Encapsulation) 🔒			
Kalıtım (Inheritance) 🧬			
Çok Biçimlilik (Polymorphism) 🎭			
Soyutlama (Abstraction) 🎭			

OOP Prensibi	Tanımı	Oyun Karakteri İçin Örneği
Kapsülleme (Encapsulation) 🔒	Verilerin korunması ve yalnızca belirli metotlar aracılığıyla erişilmesi.	Karakterin **can seviyesi** ve **enerjisi** dışarıdan değiştirilemez. Sadece belirli yetenekler veya olaylar sonucu azalır/artar.
Kalıtım (Inheritance) 🧬	Ortak özelliklerin bir üst sınıftan miras alınması.	Oyunda **Savaşçı, Büyücü, Okçu** gibi sınıflar olabilir. Tüm karakterler **temel özellikleri (can, hız, saldırı gücü)** paylaşır ama kendilerine özgü yetenekleri de vardır.
Çok Biçimlilik (Polymorphism) 🎭	Aynı metot farklı karakterler tarafından farklı şekilde uygulanabilir.	Tüm karakterler **"Saldır"** yeteneğine sahiptir ama bir **savaşçı kılıçla saldırır, okçu yayla, büyücü büyü yaparak** saldırır.
Soyutlama (Abstraction) 🎭	Karmaşık işlemlerin gizlenmesi ve yalnızca temel bilgilerin gösterilmesi.	Karakterin **"Sağlık Yenile" yeteneği** çalışır ama bunun nasıl hesaplandığını oyuncu bilmez. Oyuncu sadece "canım arttı" olarak görür.

ETKİNLİK - 1

"Bir oyun karakteri tasarlayın!" diyerek kendi sınıfımızı oluşturmak istesek adım adım nasıl olurdu ?

1.2. SINIFLAR VE NESNELER

Dünya ve çevre incelendiğinde her şeyin (cisimler, canlılar vb.) belirli özelliklerinin ve işlevlerinin olduğu görülür. Her öğrencinin bir numarası, adı soyadı, aldığı dersler gibi özellikleri ve okula gitme, sınava girme gibi işlevleri vardır. Benzer şekilde yine bir cep telefonunun rengi, boyutları, markası, adı gibi özellikleri ve çağrı başlatma, mesaj gönderme, uygulama açma gibi işlevleri bulunur.

NTP, dünyada var olan her şeyin yazılım içinde modellenmesini amaçlar. Sınıf (class), NTP'nin en önemli kavramıdır. Sınıf, nesnelerin özelliklerini ve işlevlerini (davranışlarını) tanımlamak için kullanılan bir taslaktır. Bu taslak aracılığıyla nesneler (objects) oluşturulur

Programlamada bu ev planına sınıf, ev planından yola çıkılarak yapılan gerçek eve ise nesne adı verilebilir

1.2.1 SINIF TANIMLAMA

C#'ta sınıf tanımında class anahtar kelimesi kullanılır. Aşağıda en basit sınıf tanımı verilmiştir.

```
class SinifAdi
{
}
```

Yandaki örnekte bir dikdörtgen sınıfı tanımı yapılmış ve bu sınıf içinde birkaç farklı yapı kullanılmıştır. Bu yapıların ne olduğu ve ne amaçla yazıldığı alt başlıklarda anlatılmıştır.

```
class Dikdortgen
{
private int a, b;
public Dikdortgen(int a, int b)
{
this.a = a;
this.b = b;
}
public int AlanHesapla()
{
return a * b;
}
public int CevreHesapla()
{
return 2 * (a + b);
}
}
```

1.2.2. NESNE OLUŞTURMA

Programlarda sınıfların kullanılabilmesi için bu sınıftan oluşturulan nesnelere (object) gereksinim duyulur. Bu türetme işlemine örnek oluşturma (instance) denir.

C#'ta önceden tanımlanan bir sınıftan nesne türetmek için new anahtar kelimesi kullanılır. Daha önceden oluşturulan Dikdortgen sınıfından bir nesne türetmek ve bu nesnenin ögelerini (özellikler ve metotlar) kullanmak için aşağıdaki gibi bir uygulaması yazılabilir.

```
class Dikdortgen
{
private int a, b;
public Dikdortgen(int a, int b)
{
this.a = a;
this.b = b;
}
```

```
Dikdortgen yenid = new Dikdortgen(a, b);
```

1.2.2. NESNE OLUŞTURMA

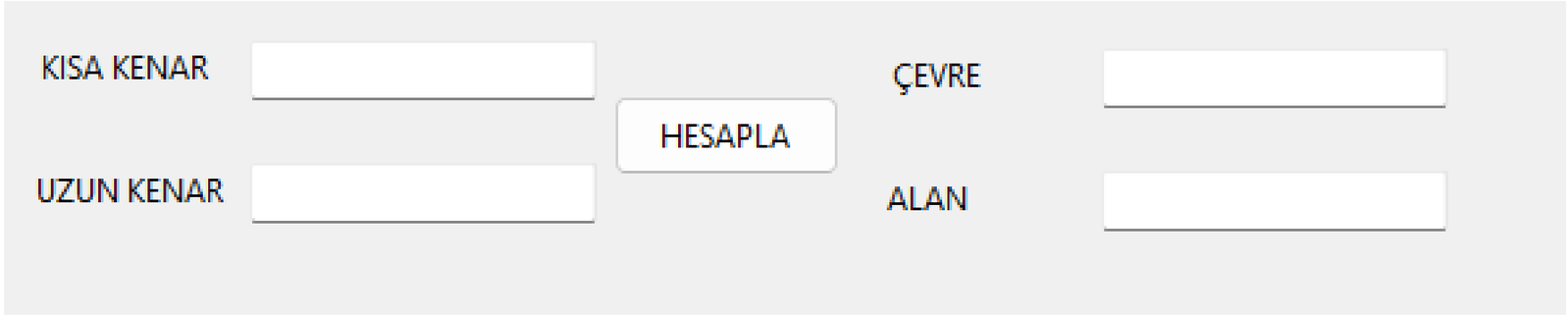
```
Dikdortgen yenid = new Dikdortgen(a, b);
```

Oluşturulan nesnenin adı “yenid” dir. Kenar uzunlukları a ve b olarak belirlenmiştir.
Nesnenin ögelerine erişmek için nokta (.) karakterinin kullanıldığı görülür. Genel bir ifadeyle yazılacak olursa nesne aşağıdaki gibi tanımlanır

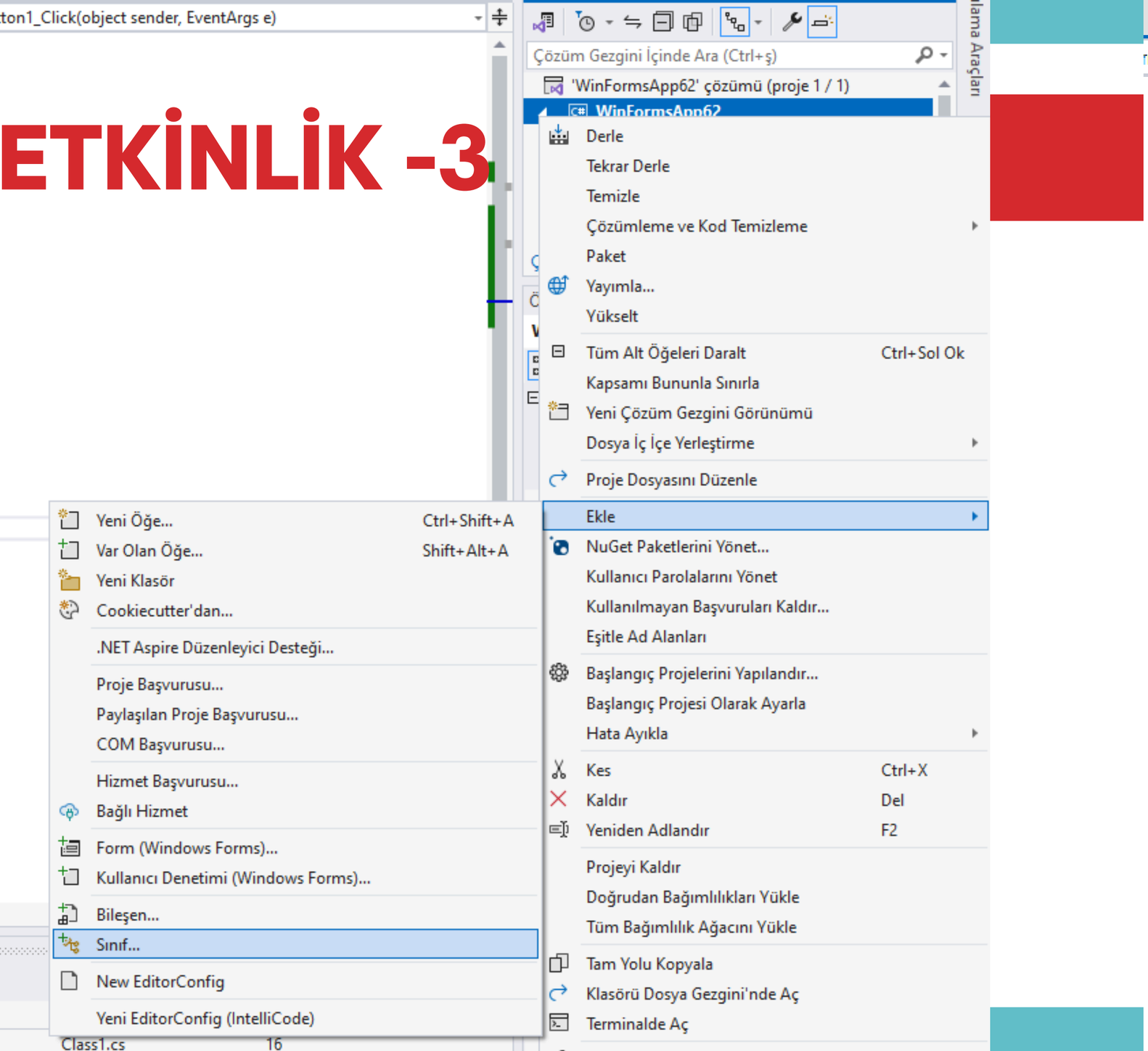
```
«Sınıf adı» «Nesne adı» = new «Sınıf adı»(««Parametre listesi»»);
```

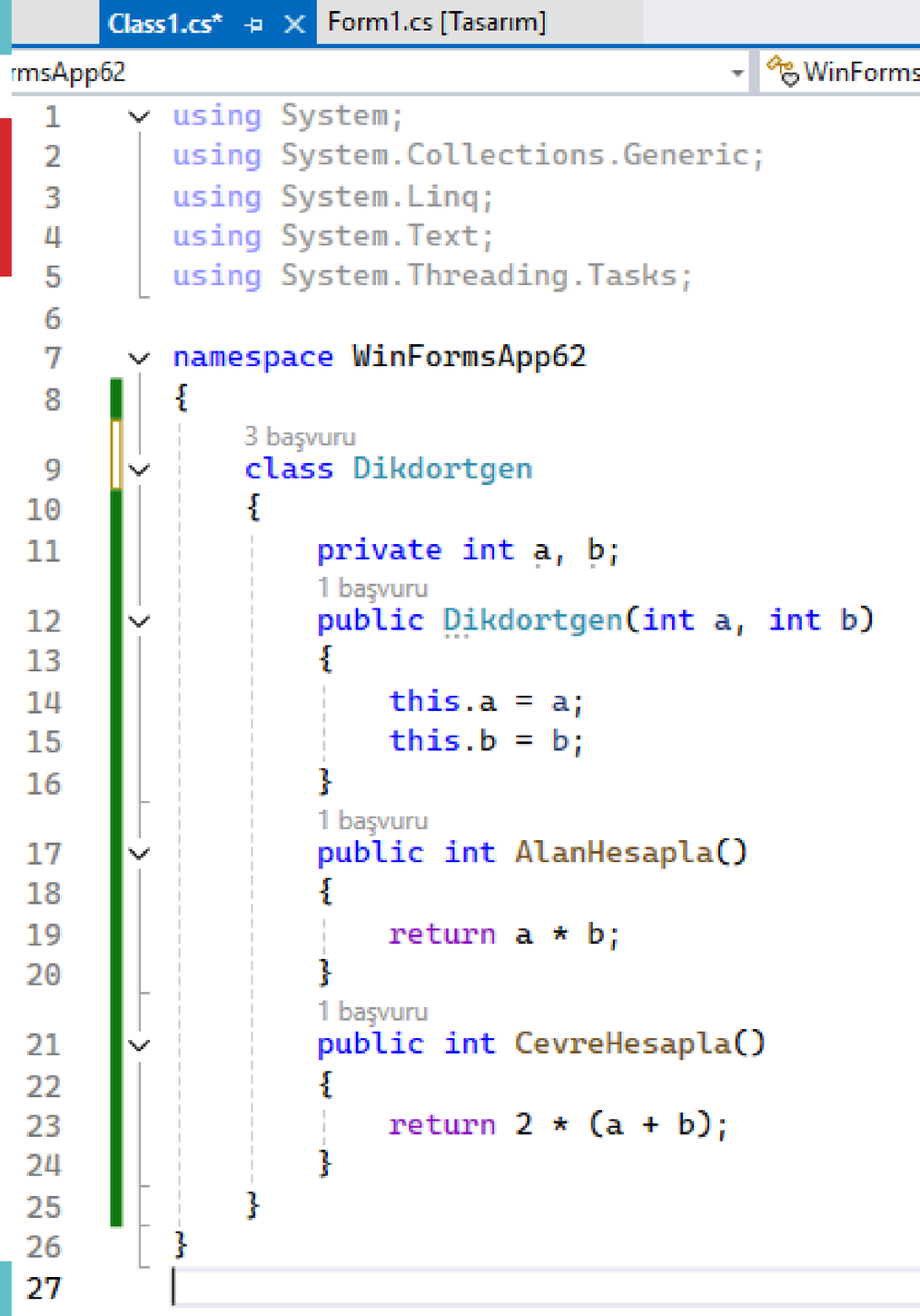
ETKİNLİK 2

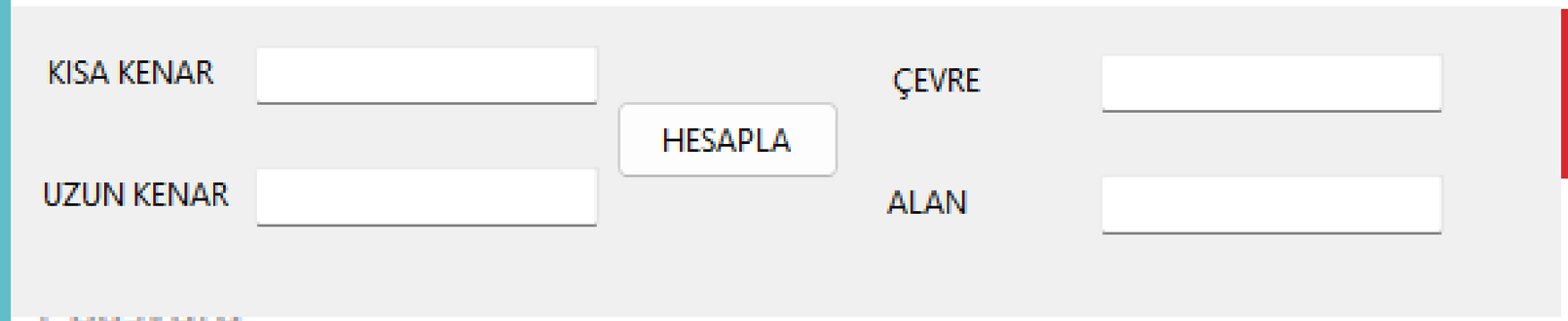
```
class Dikdortgen
{
    private int a, b;
    1 başvuru
    public Dikdortgen(int a, int b)
    {
        this.a = a;
        this.b = b;
    }
    1 başvuru
    public int AlanHesapla()
    {
        return a * b;
    }
    1 başvuru
    public int CevreHesapla()
    {
        return 2 * (a + b);
    }
}
1 başvuru
private void button1_Click(object sender, EventArgs e)
{
    int a = Convert.ToInt32(textBox1.Text);
    int b = Convert.ToInt32(textBox2.Text);
    Dikdortgen yenid = new Dikdortgen(a, b);
    int cevre = yenid.CevreHesapla();
    int alan = yenid.AlanHesapla();
    textBox3.Text = cevre.ToString();
    textBox4.Text = alan.ToString();
}
```



ETKİNLİK -3


ton1_Click(object sender, EventArgs e)
Çözüm Gezgini İçinde Ara (Ctrl+ş)
'WinFormsApp62' çözümü (proje 1 / 1)
WinFormsApp62
Derle
Tekrar Derle
Temizle
Çözümleme ve Kod Temizleme
Paket
Yayımla...
Yükselt
Tüm Alt Öğeleri Daralt Ctrl+Sol Ok
Kapsamı Bununla Sınırla
Yeni Çözüm Gezgini Görünümü
Dosya İç İçe Yerleştirme
Proje Dosyasını Düzenle
Ekle
NuGet Paketlerini Yönet...
Kullanıcı Parolalarını Yönet
Kullanılmayan Başvuruları Kaldır...
Eşitle Ad Alanları
Başlangıç Projelerini Yapılandır...
Başlangıç Projesi Olarak Ayarla
Hata Ayıkla
Kes Ctrl+X
Kaldır Del
Yeniden Adlandır F2
Projeyi Kaldır
Doğrudan Bağımlılıkları Yükle
Tüm Bağımlılık Ağacını Yükle
Tam Yolu Kopyala
Klasörü Dosya Gezgini'nde Aç
Terminalde Aç
Yeni Öğe... Ctrl+Shift+A
Var Olan Öğe... Shift+Alt+A
Yeni Klasör
Cookiecutter'dan...
.NET Aspire Düzenleyici Desteği...
Proje Başvurusu...
Paylaşılan Proje Başvurusu...
COM Başvurusu...
Hizmet Başvurusu...
Bağlı Hizmet
Form (Windows Forms)...
Kullanıcı Denetimi (Windows Forms)...
Bileşen...
Sınıf...
New EditorConfig
Yeni EditorConfig (IntelliCode)
Class1.cs 16


Class1.cs*
Form1.cs [Tasarım]
rmsApp62
WinForms
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace WinFormsApp62
8 {
3 başvuru
9 class Dikdortgen
10 {
11 private int a, b;
1 başvuru
12 public Dikdortgen(int a, int b)
13 {
14 this.a = a;
15 this.b = b;
16 }
1 başvuru
17 public int AlanHesapla()
18 {
19 return a * b;
20 }
1 başvuru
21 public int CevreHesapla()
22 {
23 return 2 * (a + b);
24 }
25 }
26 }
27



ETKİNLİK -3

```
private void button1_Click(object sender, EventArgs e)
{
    int a = Convert.ToInt32(textBox1.Text);
    int b = Convert.ToInt32(textBox2.Text);
    Dikdortgen yenid = new Dikdortgen(a,b);
    int cevre = yenid.CevreHesapla();
    int alan = yenid.AlanHesapla();
    textBox3.Text = cevre.ToString();
    textBox4.Text = alan.ToString();

}
```

Bir dikdörtgenin iki kenar uzunluğu bilgisi bulunur. Ayrıca çevre ve alan bilgilerinin hesabı da söz konusudur. Sınıf tanımında a ve b değişkenleri dikdörtgenin kenar uzunluklarını saklamak için, AlanHesapla() ve CevreHesapla() işlevleri de dikdörtgenin alan ve çevre hesabının yapılması için tanımlanmıştır.

ETKİNLİK-4

Sıra Sizde

Dikdortgen sınıfına benzer şekilde Daire sınıfını oluşturunuz.

Dairenin Yarıçapı

HESAPLA (pi = 3)

Dairenin Çevresi

Dairenin Alanı

ETKİNLİK-4

```
class Daire
{
    private int yaricap;
    1 başvuru
    public Daire(int yaricap)
    {
        this.yaricap = yaricap;

    }
    1 başvuru
    public int AlanHesapla()
    {
        return 3*yaricap*yaricap;
    }
    1 başvuru
    public int CevreHesapla()
    {
        return 2*3*yaricap;
    }
}
```

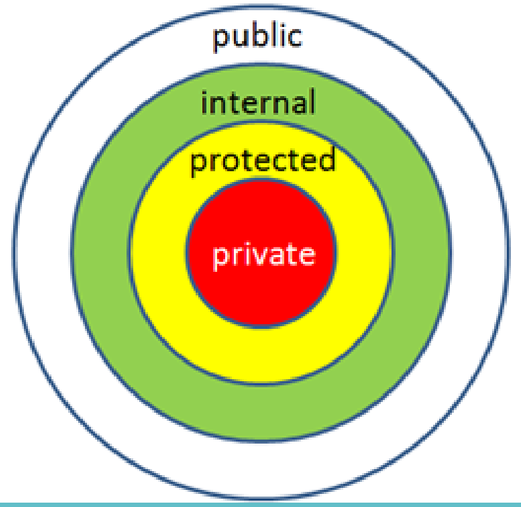
```
private void button2_Click(object sender, EventArgs e)
{
    int yc = Convert.ToInt32(textBox5.Text);
    Daire mydaire = new Daire(yc);
    int cevre = mydaire.CevreHesapla();
    int alan = mydaire.AlanHesapla();
    textBox6.Text = cevre.ToString();
    textBox7.Text = alan.ToString();
}
```

1.3. KAPSÜLLEME, ALANLAR VE

Erişim belirleyicileri ile NTP'nin temel prensiplerinden olan kapsülleme, alanlar ve özellikler aracılığıyla programlarda uygulanır. Kapsülleme, hassas veya özel bilgilerin gizlenmesi anlamına gelir.

- **Kapsülleme, verilerin korunmasını sağlar.**
- **Dışarıdan erişimi kontrol etmek için private ve public gibi erişim belirleyiciler kullanılır.**

ERİŞİM BELİRLEYİCİLER (ACCESS MODIFIERS) : .NET platformunda oluşturulan uygulamalarda güvenliği artırmak amacıyla sınıflara ve/veya sınıf içinde kullanılan ögelere erişimin kısıtlanması gerekir. Dolayısıyla koda dışarıdan erişimin sınırlarını belirlemek amacıyla erişim belirleyicileri kullanılır. C# programlama dilinde kullanılan erişim belirleyicileri şunlardır



1.3.1 ERİŞİM BELİRLEYİCİLER (ACCESS MODIFIERS)

- **public (Genel):** Public olarak tanımlanan ögeler üzerinde herhangi bir kısıtlama yoktur. Her yerden erişilebilirdir.
- **private (Gizli): En katı erişim belirleyicidir.** Ögeler sadece tanımlandığı sınıf içinde erişilebilirdir. Bir başka deyişle ögeler sadece tanımlandığı küme parantezleri arasında kullanılabilir.
- **protected (Korunumlu):** Ögeler, bulunduğu sınıf içinde ya da bu sınıftan türeyen diğer sınıflarda erişilebilirdir.
- **internal (Dâhilî):** Internal olarak tanımlanan ögelere sadece aynı program içinden erişilebilir.

- **protected internal (Dâhilî+Korumalı):** Ögeler hem protected hem de internal erişim belirleyicisine sahip olarak değerlendirilir. Türetilen sınıfın farklı program içinde olması sorun teşkil etmez

1.3.1 ERİŞİM BELİRLEYİCİLER (ACCESS MODIFIERS)

- **Bir evi düşünelim. Bu evin farklı odaları var ve her odaya herkesin girebilmesi mümkün değil! Bazı kapılar kilitli, bazıları açık, bazıları sadece aile bireyleri için erişilebilir.**

** **PUBLIC (HERKESE AÇIK) → ** Bahçe Kapısı 🌳**
  - **Bahçe kapısı herkese açık olduğu için isteyen herkes içeri girebilir.**

** **PRIVATE (YALNIZCA EV SAHİBİ ERİŞEBİLİR) → ** Kişisel Yatak Odası 🛏️**
  - **Yatak odası sadece senin girebildiğin özel bir alan.**

** **PROTECTED (AİLE ÜYELERİ ERİŞEBİLİR) → ** Evin Oturma Odası 🛋️**
  - **Oturma odası sadece evde yaşayanlar için kullanılabilir.**

** **INTERNAL (EVİN İÇİNDEKİ HERKES ERİŞEBİLİR) → ** Mutfak 🍽️**
  - **Mutfak, evde yaşayan herkes tarafından kullanılabilir, ama dışarıdan gelen misafirler erişemez.**

** **PROTECTED INTERNAL (AİLE VE EVİN İÇİNDEKİLER ERİŞEBİLİR) → ** Misafir Odası 🏡**
  - **Misafir odasına hem aile üyeleri hem de misafirler girebilir.**

Erişim Belirleyici	Günlük Hayat Benzetmesi	Kimler Erişebilir?
public	Bahçe Kapısı 🚪	Herkes erişebilir
private	Yatak Odası 🛏️	Sadece sahibi
protected	Oturma Odası 🛋️	Aile üyeleri (miras alanlar)
internal	Mutfak 🍽️	Evde yaşayanlar (proje içi)
protected internal	Misafir Odası 🏡	Evde yaşayanlar + aile dostları

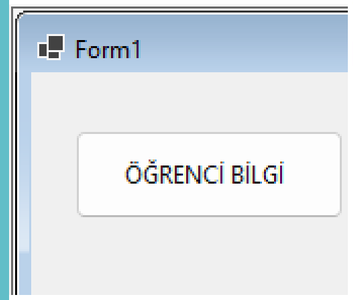
1.3.2. ALANLAR (FIELDS)

Bir alan (field), bir sınıfın içinde bulunan ve o sınıfın özelliklerini, herhangi türden (int, string vb.) saklayan bir değişkendir.

```
class Ogrenci  // Öğrenci sınıfı
{
    // ALANLAR (FIELDS)
    public string Ad;  // Öğrencinin adı
    public int Numara; // Öğrencinin numarası
    public string Sinif; // Öğrencinin sınıfı
}
1 başvuru
private void button3_Click(object sender, EventArgs e)
{
    // Bir öğrenci nesnesi oluşturalım
    Ogrenci ogr1 = new Ogrenci();
    ogr1.Ad = "Ali";
    ogr1.Numara = 1203;
    ogr1.Sinif = "7A";

    // Öğrencinin bilgilerini ekrana yazdıralım
    MessageBox.Show("Öğrenci Bilgileri:");
    MessageBox.Show("Ad: " + ogr1.Ad);
    MessageBox.Show("Numara: " + ogr1.Numara);
    MessageBox.Show("Sınıf: " + ogr1.Sinif);

}
```

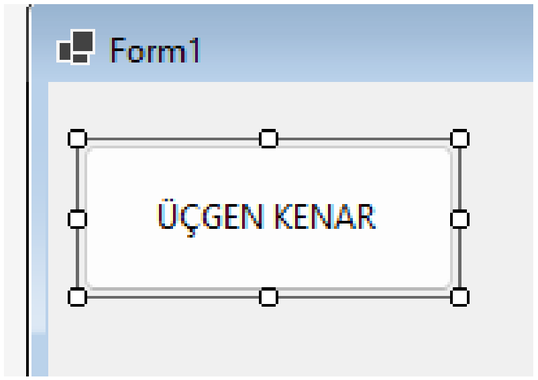


Sınıfa ait alanlar tanımlanırken başına “public” erişim belirleyicisi yazılmıştır.
Bu erişim belirleyicisi, alan bilgisine sınıf dışından erişim için gereklidir. Alanlar yalnızca özel ve gizli kalması gereken değişkenler için kullanılmalıdır.
Sınıf içinde tanımlanmış bir değişkenin başına yazılan “public” erişim belirleyicisi ile alanı dış dünyaya açmak uygun değildir.

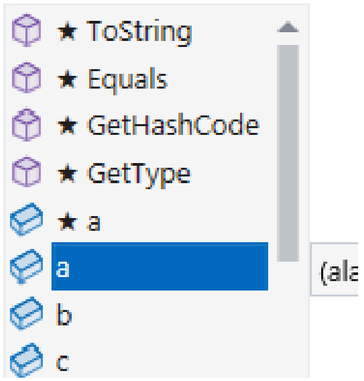
Bu şekilde yapıldığında değişkene değer atama ya da değişkenin değerinin okunması işlemlerinde kontrol mekanizması işletile

ETKİNLİK-5

Butona basınca daha önceden tanımlanan kenarlar mesaj olarak ekranda gözüksün




```
class Ucgen
{
    public int a;
    public int b;
    public int c;
}
1 başvuru
private void button4_Click(object sender, EventArgs e)
{
    Ucgen ucgen = new Ucgen();
    ucgen.a = 3;
    ucgen.b = 4;
    ucgen.c = 5;
    MessageBox.Show("Üçgenin a kenarı  " + ucgen.);
}
```



ETKİNLİK-6

Aşağıdaki örnek alan ve sınıf tanımlarını kodlayınız. sizde iki tane farklı konuda sınıf ve alan tanımlayınız....

```
class Araba
{
    // ALANLAR (FIELDS)
    public string Marka;
    public string Model;
    public int Hiz;
}

class Program
{
    static void Main()
    {
        // Bir araba nesnesi oluşturalım
        Araba araba1 = new Araba();
        araba1.Marka = "Toyota";
        araba1.Model = "Corolla";
        araba1.Hiz = 180;
```

```
class Karakter
{
    // ALANLAR (FIELDS)
    public string Ad;
    public int Can;
    public int SaldiriGucu;
}

class Program
{
    static void Main()
    {
        // Bir oyun karakteri oluşturalım
        Karakter karakter1 = new Karakter();
        karakter1.Ad = "Ejderha Avcısı";
        karakter1.Can = 100;
        karakter1.SaldiriGucu = 50;
```

1.3.3. ÖZELLİKLER (PROPERTIES)

Bir özellik (property), bir sınıfın içindeki değişkenlere (alanlara) güvenli bir şekilde erişmemizi sağlar.Bu sayede, NTP'nin temel prensiplerinden "kapsülleme" prensibi sınıfa uygulanır.

Bir telefonun **ses seviyesini** değiştirmek istiyorsunuz. Ama telefonun **iç donanımına doğrudan erişemezsiniz!**

💡 **İşte burada bir "özellik" devreye girer.** Telefonun **ses açma ve kapama butonları** vardır.

- **Ses seviyesini artırmak için butona basarsınız** 🎚️
- **Ama telefonun iç devrelerini göremezsiniz!**

📌 **Telefonun Ses Ayarı Nasıl Bir Property ile Anlatılabilir?**

Özellik (Property)	Ne İşe Yarar?
SesSeviyesi	Telefonun ses seviyesini artırır veya azaltır.
EkranParlaklığı	Ekranın parlaklığını ayarlar.
BataryaSeviyesi	Bataryanın yüzdesini gösterir.

Bu özellikler sayesinde telefonun **donanımına doğrudan dokunmadan ayar yapabiliriz!** 📱🔧

1.3.3. ÖZELLİKLER (PROPERTIES)

Bir arabanın **hızını değiştirmek** istiyorsunuz. Ama motorun içine girip kabloları çekemezsiniz!

💡 **Bunun yerine "Gaz Pedalı" kullanırsınız.**

- Gaza bastıkça hız artar. 🏎️
- Frene bastıkça hız azalır. 🛑

📌 **Arabanın hızını ayarlamak için bir property (özellik) kullanabiliriz!**

Özellik (Property)	Ne İşe Yarar?
Hiz	Arabanın hızını gösterir ve ayarlamamızı sağlar.
YakıtMiktari	Arabanın deposundaki yakıtı gösterir.
MotorDurumu	Motorun çalışıp çalışmadığını gösterir.

💡 **Burada "Hiz" özelliği sayesinde arabanın hızını kontrol edebiliyoruz!**

1.3.3. ÖZELLİKLER (PROPERTIES)

Get (almak, elde etmek) ve set (düzenlemek, ayarlamak) şeklinde iki ayrı alt metodu bulunur.

get Metodu: Bir değer döndürmek için kullanılır. Özelliklerin get metodunda return anahtar kelimesi kullanılarak *"return…;"* ile bir değerin döndürüleceği belirtilir.

set Metodu: Değişkene değer atama işlemleri için kullanılır. Burada görülen *value* anahtar kelimesi dışarıdan bu özelliğe gönderilen değeri temsil eder.

Özellikler (Properties), sınıflardaki özel alanlara (fields) güvenli bir şekilde erişmemizi sağlar.

📌 Get ve Set metotları, bu özelliklerin nasıl okunacağını ve değiştirileceğini kontrol eder.

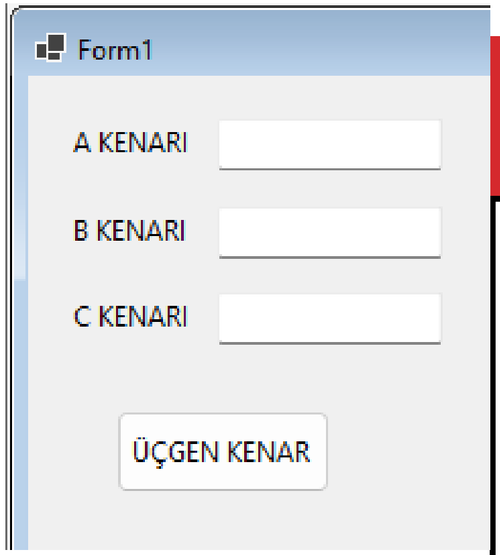
Bir banka hesabınız olduğunu düşünün.

- Banka hesabınızdaki para miktarı gizlidir (private alan).
- Bu miktarı öğrenmek isterseniz bir banka uygulamasına giriş yaparsınız (get metodu).
- Hesaba para yatırmak veya çekmek için bir işlem yapmalısınız (set metodu).

💡 Get ve Set Metotları, banka hesabına erişimi kontrol eden işlemler gibidir!

Özellik (Property)	Ne İşe Yarar?
get	Hesaptaki para miktarını gösterir.
set	Hesaba para yatırmanıza veya çekmenize izin verir.

ETKİNLİK-7




```
class Ucgen
{
    public int a;
    public int b;
    public int c;
    2 başvuru
    public int A
    {
        get { return a; }
        set
        {
            if (value <= 0)
                MessageBox.Show("Hatalı Bilgi");
            else
                a = value;
        }
    }
    2 başvuru
    public int B
    {
        get { return b; }
        set
        {
            if (value <= 0)
                MessageBox.Show("Hatalı Bilgi");
            else
                b = value;
        }
    }
    2 başvuru
    public int C
    {
        get { return c; }
        set
        {
            if (value <= 0)
                MessageBox.Show("Hatalı Bilgi");
            else
                c = value;
        }
    }
}
1 başvuru
private void button1_Click(object sender, EventArgs e)
{
    Ucgen ucgen = new Ucgen();
    ucgen.A = Convert.ToInt32(textBox1.Text);
    ucgen.B = Convert.ToInt32(textBox2.Text);
    ucgen.C = Convert.ToInt32(textBox3.Text);
    MessageBox.Show("Üçgenin A kenarı   : "+ucgen.A.ToString());
    MessageBox.Show("Üçgenin B kenarı   : "+ucgen.B.ToString());
    MessageBox.Show("Üçgenin C kenarı   : "+ucgen.C.ToString());
}
```

ETKİNLİK-7

```
{
    if (value <= 0)
        MessageBox.Show("Hatalı Bilgi");
    else
        a = value;
}
```

Yandaki kod satırları ile dışarıdan alınan bilginin kontrolü gerçekleştirilir, sıfır veya negatif bir değer gönderildiğinde kullanıcıya hata mesajı gösterilir