SINIFLAR(CLASS)
OOP (Object Oriented Programming)
Nesneye Yönelik Programlama

Arif GÜNEL

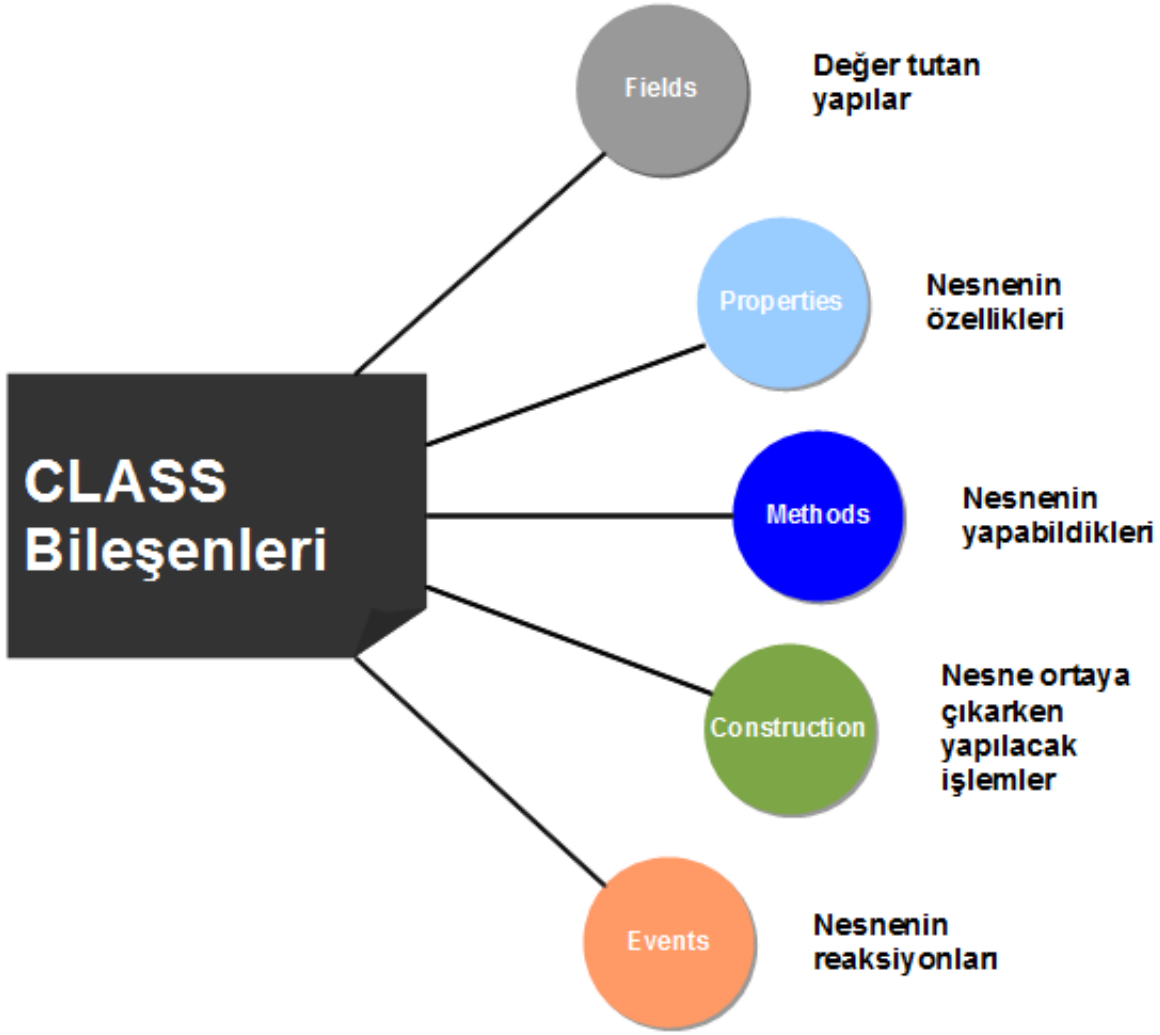
Nesneye Yönelik Programlama

- *Object-oriented programming yani nesne yönelimli programlamaya giriş sınıf kavramlarının anlaşılması ile başlamaktadır.*
- *Çünkü **sınıflar** nesne yönelimli programının **temel yapı taşıdır**.*
- ***Sınıf (class)** bir nesnenin şeklini tanımlayan bir şablondur.*

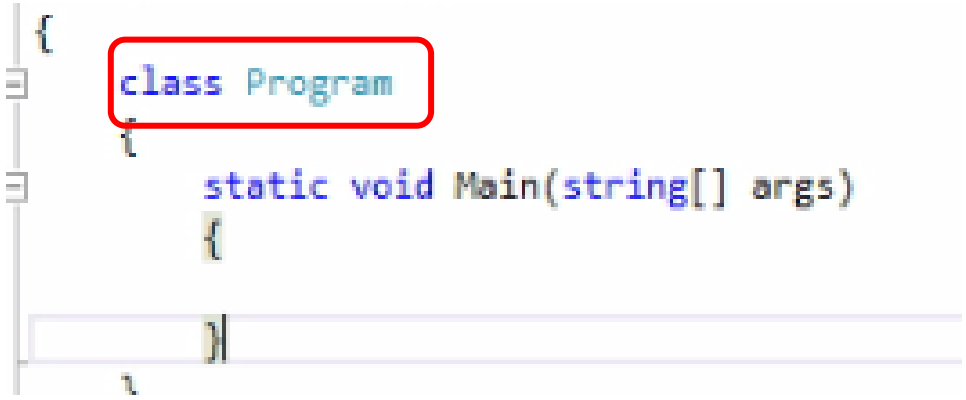
- *Sınıflar, verileri ve bu verileri işleyecek kodu içerir. Sınıf kendisinden türeyecek olan nesneleri oluşturmak için gerekli olan temel özellikleri içerir.*
- ***Nesne(object)** 'ler sınıfların birer örneğidirler.*
- *Bu durumda sınıf, nesnelerin nasıl inşa edileceğini tanımlayan bir kılavuzdur diyebiliriz.*

- *Sınıflar sayesinde programlar parçalara bölünür ve karmaşıklığı azalır.*
- *Oluşturulan metodlar ve özellikler bir sınıfın içerisinde yer alır ve bir sınıf defalarca kullanılabilir.*
- *Bir sınıfta hem fonksiyonlar hem de veriler aynı anda birbiriyle sıkı bir şekilde bağlı olarak bulunurlar.*

- *Bir sınıf kendisinde oluşturulacak nesneler için bir takım üyeler içermelidir.*
- *Bu üyeler; alanlar (fields), metodlar (methods), yapıcılar (constructor), özellikler (properties), olaylar (events), delegeler (delegates), vb...dir.*
- *Sınıf nesneler için bir şablon görevi görmektedir.*
- *Yani sınıf nesnelerin durumları ile ilgili işlemleri ve özellikleri tanımlar. Ortak özelliklere sahip nesnelere ait veri ve yordamlar bir sınıfın içinde toplanır. Bu sınıf yapısı kullanılarak programın içinde nesneler tanımlanır.*



- *Yazdığımız programlar bile bir sınıf biz bu sınıfın altına tanımladığımız metot vb argümanlar ile projemizi oluşturuyoruz.*



- *Programlama dillerinde nesne kavramı günlük hayatta fiziksel olarak bir takım özelliklerini niteleyebildiğimiz eşyaların programlama dillerindeki karşılıklarına sınıf ya da nesne diyoruz.*
- *C#' da sınıf tanımlarken* **class** *anahtar kelimesini kullanıyoruz.*

Nesne yönelimli programlama tekniğini açıklayacak olursak;

- *“Çözülmesi istenen problemi temel parçalara bölüp sonra bu parçalardan aralarında benzerlik ve ilişki bulunan başka parçalar üretip büyük projeler gerçekleştirme yöntemine nesne yönelimli programlama denir.”*
- *Nesne yönelimli programlamada kullanılan her şey bir nesnedir.*
- *C# %100 bir nesne yönelimli programlama dilidir*

***Nesnelerin önemli iki özelliği vardır**;*

1. *Durum*
2. *Davranış*

örneklendirelim

- *Mesela bir telefon düşünelim. Bu telefonun rengi, fiyatı, markası, modeli, boyutları, işlemcisi, hafızası vs. gibi özellikleri onun durumunu belirtir. Buna attribute (özellik) de denir. Yani biz nesnemizin durumlarını değişkenler ile tanımlayacağız.*
- *Yine bu telefonun çağrı yapmak, resim çekmek, ses kaydı almak, parlaklık ayarını değiştirmek gibi eylemleri ise onun davranışını belirtir. Dolayısıyla buradan şöyle bir tanım çıkartabiliriz. Yani biz nesnemizin durumlarını değişkenler ile tanımlayacağız.*

Nesne yönelimli programlamaya günlük hayattan örnek vererek açıklayalım;

- *Mesela araba aldığımızda bu arabanın marka ve modelden bağımsız olarak bir takım özelliklerinin standart olduğunu biliriz.*

Nedir bu özellikler;

- Rengi,
- Fiyatı,
- Markası,
- Modeli,
- Beygir gücü,
- Yakıt tipi,
- Ağırlığı

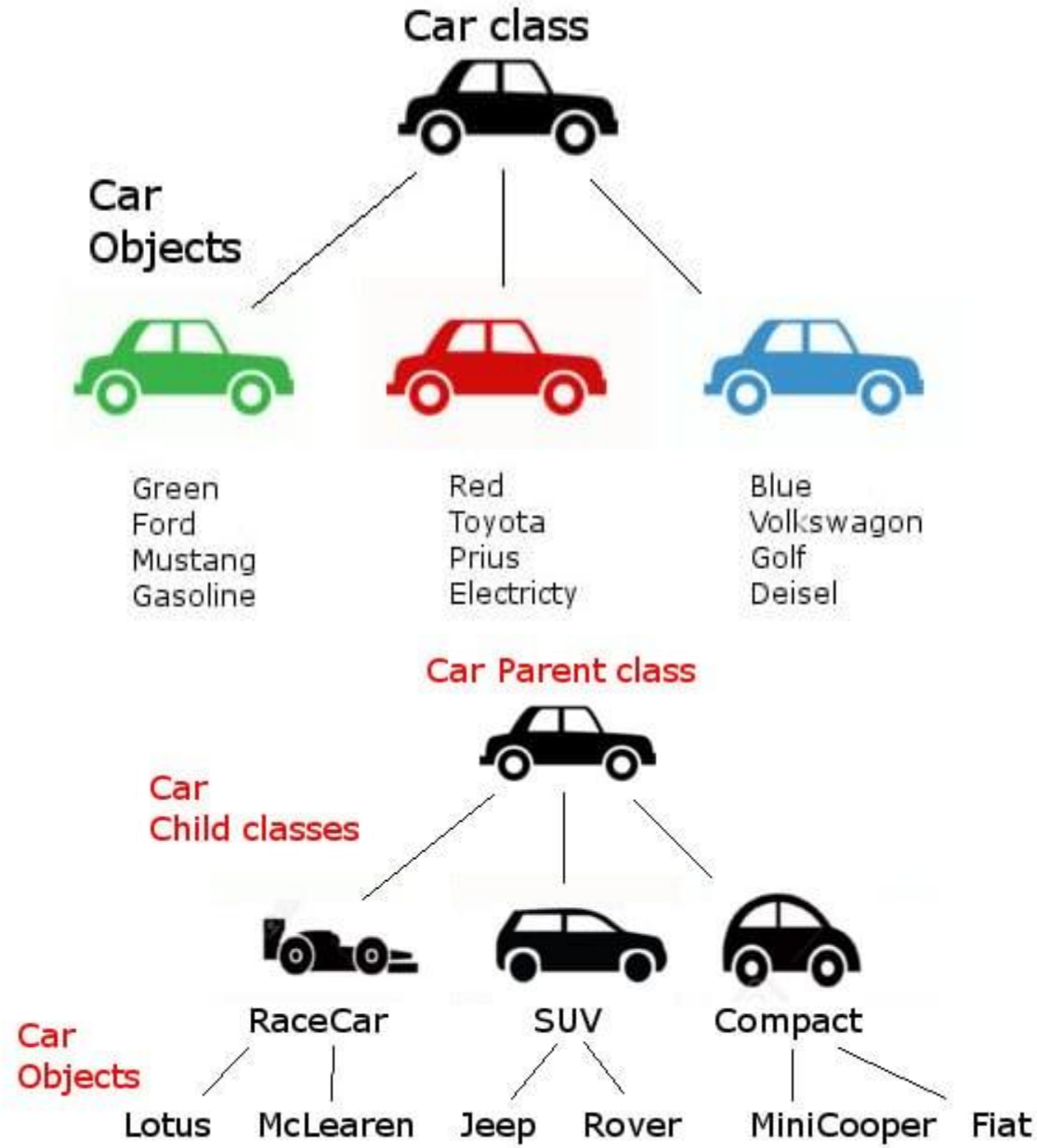
- *Bir yakıt türü(benzin, LPG, Mazot, Elektrik) kullanıp bu sayede enerji elde eden, bu enerji ile dönme hareketi sağlayıp bu sayede ilerleyen, fren, motor, kapı, vites gibi, vb….. özellikleri olur*

Davranışlar;(metotlarımızı temsil eder)

- *Hızlanması,*
- *Yavaşlaması,*
- *Vites değiştirmesi,*
- *Komuta edilmesi*

- *Her araba markası bu temel özellikleri sağlar bu özelliklerin bazılarını alır ve üzerine bir takım özellikler ilave eder.*
- *Bizde yazılım tarafında bu sınıfın özelliklerini kullanarak farklı marka ve modellerde araç nesneleri türetiriz.*

- *Her araba markası bu temel özellikleri sağlar bu özelliklerin bazılarını alır ve üzerine bir takım özellikler ilave eder.*



- *İşte nesne yönelimli programlama temel olarak aralarında ilişki bulunan nesnelerin temel tanım ve özelliklerinin alınarak bu özelliklerinin üzerine yeni prosedür ve özelliklerin eklenmesi ve bu eklentiler ile uyum içerisinde çalışması mantığı ile çalışmaktadır.*
- *Nesne yönelimli ve sınıf mantığı arabanın herhangi bir özelliğine ulaşmamız için önce arabaya sahip olmamız gerekmesi gibi. Ve araba nesnesini aldığımızda arabanın tüm özelliklerine sahip olmak isteriz. Bu programlama tarafından nesne üzerinden işlem yapma miras alma örnekleme gibi kavramlar ile karşımıza çıkmaktadır.*

- *Java, C++, C#, Python, PHP, JavaScript, Ruby, Perl, Smalltalk, Objective-C gibi diller başlıca nesne yönelimli programlama dilleridir*
- *Nesnelerin programlamadaki karşılığı sınıflardır(class).*
- *Bu hafta sınıfların mantığını anlamaya çalışacağız*

Nesne yönelimli Programlamanın Dört Temel Özelliği

- *Nesne yönelimli Programlamada gerçekleştirilmesi beklenen dört temel özellik belirlenmiştir. Eğer bu dört temel özellikten birinin bile eksik olması o programlama dilinin nesne yönelimli sayılmamasına neden olur.*
- *Bu dört özellik;*
  - *Encapsulation(Kapsülleme -sarma-kuşatma)*
  - *Inheritance (miras - kalıtım)*
  - *Polymorphism(Çok biçimlilik- çok şekilcilik)*
  - *Abstraction(soyutlama- çıkarma - ayırma)dır.*

Encapsulation(Kapsülleme -sarma-kuşatma)

Genel tanımıyla kullanıcı tarafından verilerin, sınıfların ve metotların ne kadarının görüntülenebileceği ve değiştirilebileceğinin sınırlarının konulmasını sağlar.

üç adet access modifier'dan (erişim dönüştürücüsü) bahsedilebilir.

Bunlar;

- *Public (herkese açık),*
- *protected (koruma altında)*
- *private (özel)*

- ***Public*** *olanlar herkes tarafından görülebilir ve değiştirilebilir yani en güvensiz sınıf çeşididir. Bir program yazılırken programın iç yapısını değiştirecek metotların Public olması önerilmez. Public modifier dış kullanıcı tarafından eklenmesi veya değiştirilmesi istenen veriler için kullanılır.*

- ***Protected*** *modifier, public modifier'dan daha güvenli bir access modifier'dır. Aynı sınıf içinde ve üst sınıflar, ondan türetilmiş sınıflar ve aynı paket içinde bulunan sınıflar tarafından görüntülenebilir veya erişebilirler.*

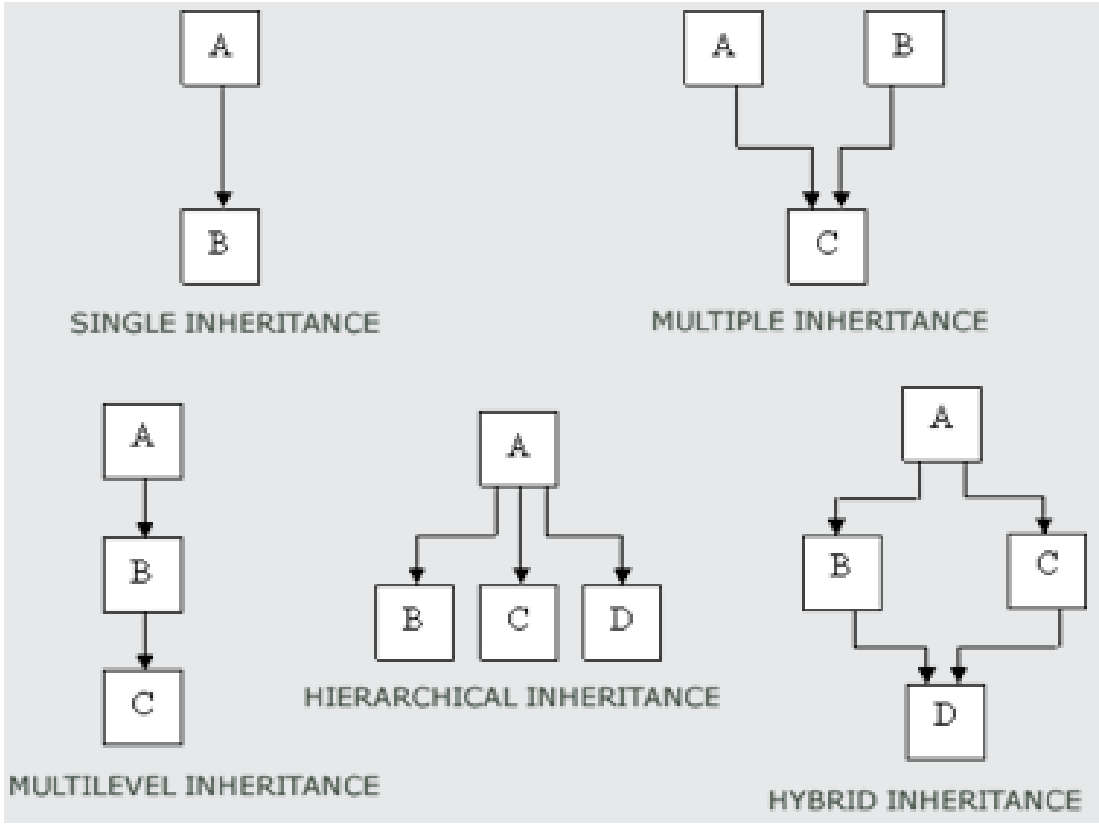
- ***Private*** *en güvenli access modifier'dır. Private olanlar yalnızca içinde olduğu sınıf tarafından görülebilir veya erişilebilirler. Sınıflar private olabileceği gibi özellikleri ve üstünde tuttuğu veriler de private olabilir.*

Inheritance (Miras alma- Kalıtım)

- *Bir **sınıftan** başka bir **sınıf** türetirken aralarında bir **alt-üst ilişkisi** oluşturmayı ve bu **sınıflar üzerinde ortak metotlar ve özellikler kullanılmasını sağlayan** bir mekanizmadır.*
- *Nesne yönelimli programlamadaki en temel kavramlarından biridir.*
- *Hali hazırda var olan sınıfların üzerine başka sınıfların inşa edilmesini sağlar.*

*5 çeşit **Inheritance** çeşidinden söz edilebilir.*

- ***Single Inheritance (Tekli Kalıtım):** Alt sınıf tek bir üst sınıfın tüm özelliklerini taşır.*
- ***Multiple Inheritance (Çoklu Kalıtım):** Bir alt sınıf birden fazla üst sınıfın tüm özelliklerini taşır.*
- ***Multilevel Inheritance (Çok Seviyeli Kalıtım):** Bir sınıfın alt sınıfı oluşturulduktan sonra bu alt sınıfın da bir alt sınıfının oluşturulmasına denir.*
- ***Hierarchical Inheritance (Hiyerarşik Kalıtım):** Bir üst sınıfın birden fazla alt sınıfa base class (temel sınıf)'lik yapmasına denir.*
- ***Hybrid Inheritance (Melez Kalıtım):** Öbür Inheritance türlerinin 2 veya daha fazlasını barındıran Inheritance türüdür.*



Polymorphism (çok biçimlilik)

- *Bir sınıfı başka bir sınıfatan miras alıp oluşturduktan sonra miras alınan bazı özellikleri değiştirmesine çok biçimlilik özelliği denir.*
- *Programlama dilinin, **metotları** ve **türetilmiş sınıfları** yeniden tanımlama yeteneğidir.*
- `Virtual` *ve* `override` *tanımlamaları önemlidir.*

```
public class Shape
{
    // A few example members
    public int X { get; private set; }
    public int Y { get; private set; }
    public int Height { get; set; }
    public int Width { get; set; }

    // Virtual method
    public virtual void Draw()
    {
        Console.WriteLine("Performing base class drawing tasks");
    }
}

public class Circle : Shape
{
    public override void Draw()
    {
        // Code to draw a circle...
        Console.WriteLine("Drawing a circle");
        base.Draw();
    }
}
public class Rectangle : Shape
{
    public override void Draw()
    {
        // Code to draw a rectangle...
        Console.WriteLine("Drawing a rectangle");
        base.Draw();
    }
}
```

Abstraction (soyutlama)

- ***Soyutlama, iç detayları gizleyerek sadece işlevleri göstermeye denir***

- ***Alt sınıfların*** *ortak özelliklerini ve işlevlerini taşıyan ancak henüz bir* ***nesnesi*** *olmayan bir* ***üst sınıf*** *oluşturmak istenirse bir* ***soyut (abstract) üst sınıf*** *oluşturulur.*

- ***Soyut sınıfın yöntemleri alt sınıfları*** *tarafından üzerine yazılmak üzere şablon olarak tanımlanabilir veya* ***soyut metot*** *olarak oluşturulabilir.*

- ***Soyut metota*** *sahip bir sınıf otomatik olarak kendisi de* ***soyut*** *hale gelir ve* ***soyut sınıflardan nesne*** *oluşturulmaz*

- *“Sekil” adlı bir soyut sınıf oluşturulmuştur.*
- *Bu sınıfa alt sınıf olarak “Dikdortgen” diye gerçek bir sınıf oluşturulmuştur.*
- *Bu alt sınıfa “cevre” diye bir metot oluşturulmuştur ve “Sekil” altındaki öbür alt sınıflar ile ortak bir metot olduğu için bu metot “Sekil” soyut sınıfının altına da eklenmiştir.*

```
public abstract class Sekil
{

    public int cevre()
    {
        return 0;
    }
}


public class Dikdortgen : Sekil
{
    private int boy;
    private int en;

    public int cevre()
    {
        return (2 * (boy + en));
    }
}
```

Sınıf mantığı üzerinden programcılığa bakış

- *Örneğin bir araba satışı yapan firma olduğunu hayal edelim.*
- *Bizden genel süreçlerini takip edeceği bir yazılım oluşturmamızı istedi.*
- *Bu durumda ilk olarak firmanın ihtiyaçlarını ve isteklerini ve gelecekte ortaya çıkabilecek ona özgü durumları analiz ederek işe başlanmaktadır.*

Genel anlamda satış yapma işinde;

- *Alıcı : Müşteri*
- *Satıcı : Personel(Satış sorumlusu)*
- *Satılacak nesne : Araç(araba, iş makinası, motor, kamyon vb.)*
- *Diğer : Fatura, Kaparo alıp ayırma, sipariş, vb. ihtiyaçlar)*

Burada tanımlanan her başlık bir nesnemiz olacak. Ve bu nesnelerin temel özellikleri- işlevselliği(davranışı) olmak zorunda.

- ***Alıcı(Müşteri)*** ***:*** *TC. No, Ad, Soyad, Tel No, Adres, Cinsiyet, Adres ……*
- ***Personel*** ***:*** *Görev türü(satış temsilcisi, temizlik per,), sicil no, ad, soyad, TC.No ,Cinsiyet, Tel No, Adres*

Burada dikkat edilirse müşteri ve personel nesnelerinin birkaç özelliği hariç diğer alanlarının ortak olduğu görülmektedir. Bu durum nesne yönelimli programlamanın miras alma-kalıtım özelliğinde inceleyeceğiz.

- ***Araç*** ***:*** *Marka, model, yakıt, km,*
- ***Sipariş*** ***:*** *Müşteri No, Fiyat, vb*

Bu kısımda sipariş nesnesinde diğer nesnelere erişmek(nesne içerisinde nesneye erişmek) istediğimizde referans gösterme işlemini ele alacağız.

Örnek üzerinden inceleme yapalım

- *Bu örneğimizde sınıf tanımlama ve tanımladığımız sınıflardan yeni nesneler üretmeyi inceleyeceğiz.*
- ***Yeni bir proje oluşturalım;***
- *Öncesinde biz şimdiye kadar* `class Program` *isminde sınıfı kullandık ve bu sınıfın altında main metodu ya da bizim tanımladığımız özel metotları kullandık.*
- *Şimdi ise sınıflar tanımlayıp bu sınıflardan nesneler oluşturmayı yapacağız. Nesneleri nasıl kullanıldığından bahsedeceğiz.*
- *Sınıflara günlük hayatta kullandığımız nesnelerin programlama dilindeki karşılığı denebilir.*

Sınıf tanımlanırken `class` *anahtar kelimesi ile yapıyoruz.*

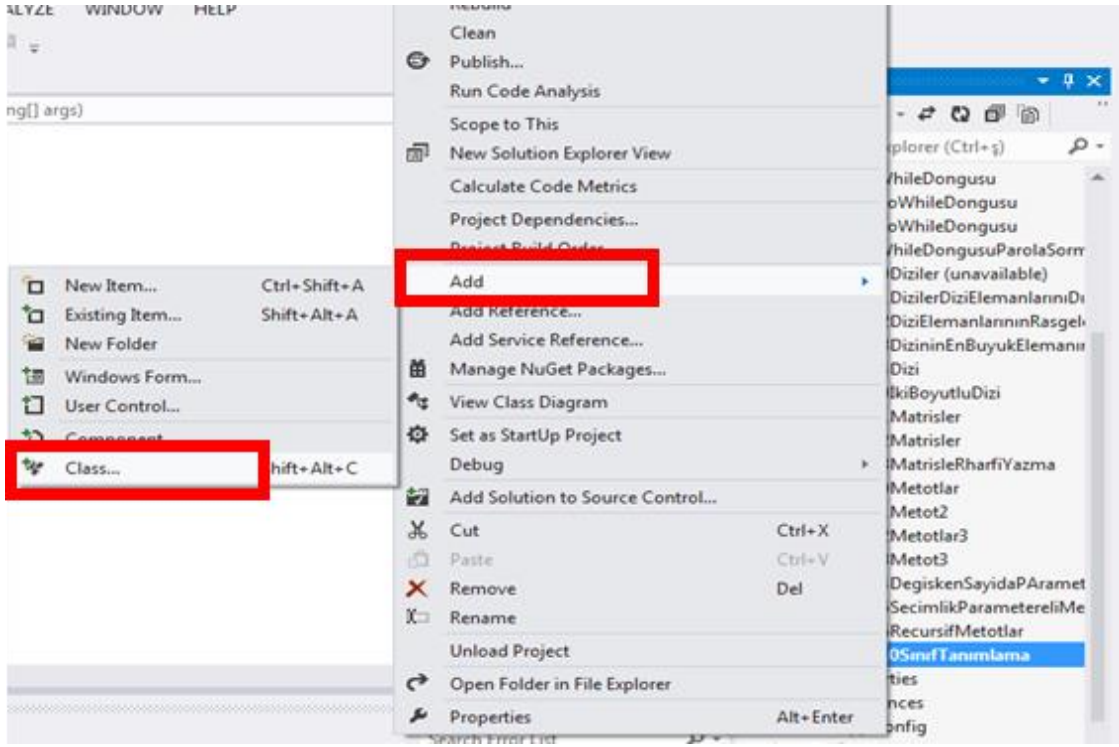
Projemize yeni bir sınıf eklerken;

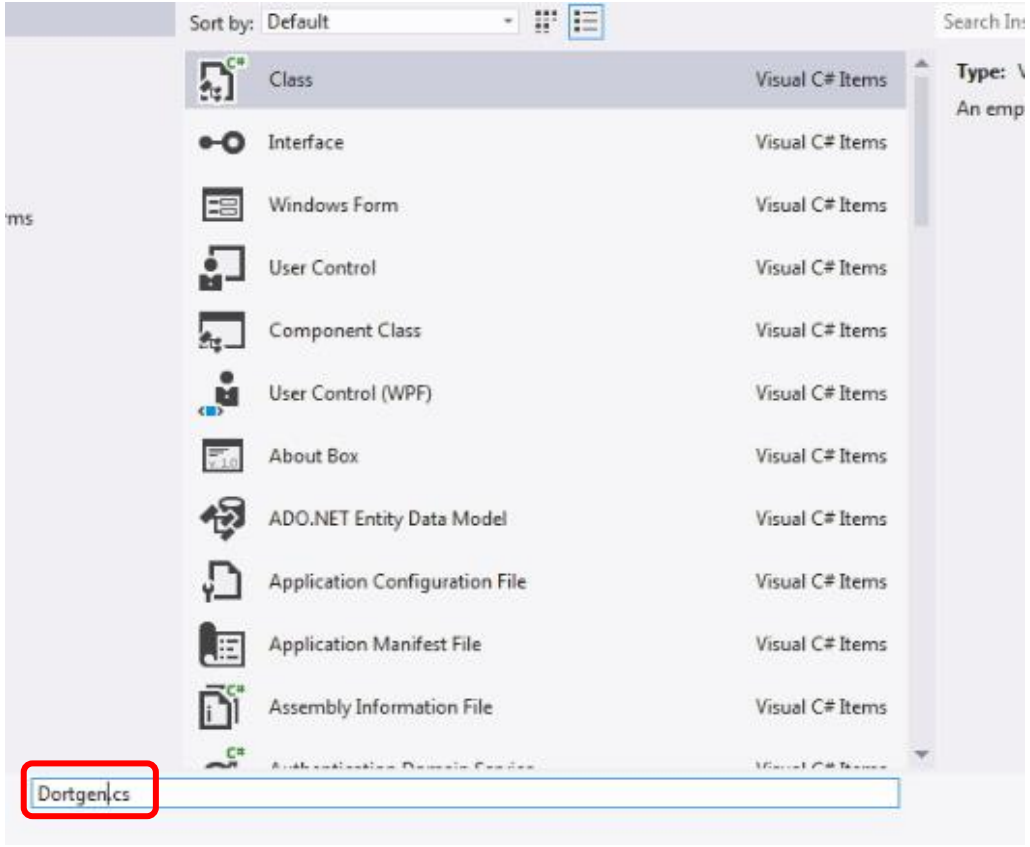
- `namespace` *in içerisinde tanımlayabileceğimiz gibi*
- *ya da* ***ayrı bir dosya ekleyerek*** *bu kısımda sınıf oluşturulabilir.*

Genelde projelerde ayrı sınıflar ayrı dosyalarda tutulması tercih edilebilir. Çünkü tüm sınıfların aynı dosyada tutulması karmaşaya neden olabilir.

- *Bizde projemize yeni bir class dosyası ekleyelim ve sınıfımızı ayrı olarak tutalım;*

- *Sınıfımıza **Dortgen** ismini verelim.*





- *Class'ımızı yani sınıfımızı* `public` *anahtar kelimesi ile tanımlıyoruz. Public anahtar kelimesi bir erişim belirtecidir.*
- *Şimdilik açıklamayalım ileride inceleyeceğiz.*

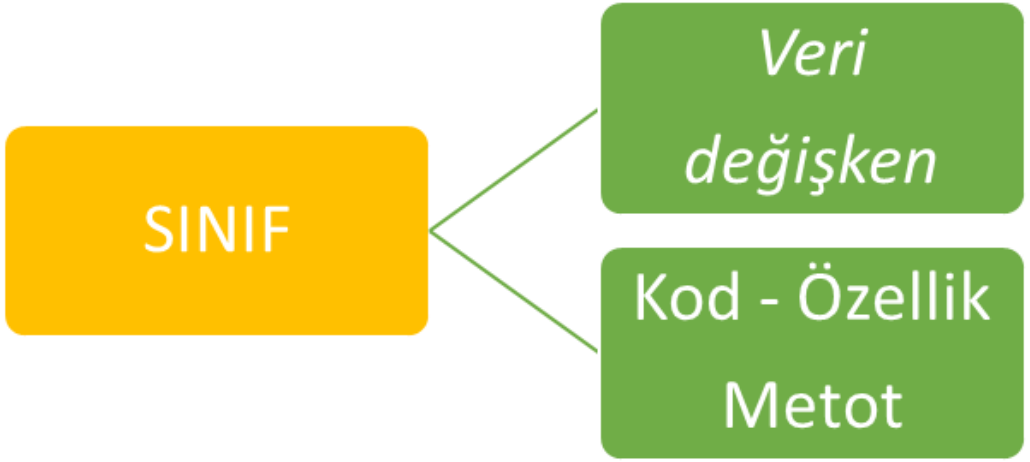
```
public class Dortgen
```

- *dediğimizde Dörtgen isminde bir sınıf tanımlamış oluyoruz.*
- *Bu Dörtgen sınıfı içerisinde yeni metotlar tanımlayabiliriz, dörtgenlerin temel özelliklerini tanımlayabiliriz.*

- *Bu kısımda bir çok örnek verilebilir;*

- *mesela sandalye class'ı desek temel olan bir sandalyenin özelliklerini tanımlarız. Neler olabilir 4 ayağı(uzunlukları kanlığı gibi), sırtımızı yasladığımız kısım, yapıldığı malzeme(ahşap, demir vb), deseni,rengi gibi…*

- *Banka hesabı örneği(hesap isminde sınıfımız olabilir. Bu sınıf içerisinde hesap sahibi adı, hesap no, kredi* limiti, iban gibi)



- *Sınıf soyut bir ifadedir, nesneler oluşuncaya kadar fiziksel olarak bellekte yer almazlar. Sınıfta veriler veri üyelerinde(değişken), kod ise fonksiyon üyelerinde saklanır.*
- *Veri üyeleri örnek değişkenleri ve statik değişkenleri içerir. Fonksiyon üyeleri ise metot, yapılandırıcı, yok edici, indeksleyici, olaylar, operatörler ve özellikleri içerirler*.

Örneğimize dönelim;

Dörtgen sınıfında hangi özellikler olabilir diye düşünüyoruz. Yani farklı dörtgenler oluşturmak istediğimizde bu oluşturacak olduğumuz dörtgenlerin nelere ihtiyacı olabilir diye düşündüğümüzde;

- *en, boy, alan, çevre uzunluğu olabilir.(en ve boy özellik, alan ve çevre hesaplama metot olacak)*
- *Bu sınıfın içerisinde en ve boy olarak iki özellik tanımlanabilir.*

Değişken tanımlama kalıbı;

önce	*sonra*	*en sonunda ise*
erişim türünü*(`public`),*	***veri tipini****(`int`)*	***değişken ismini*** *(En)*

erişim türünü veri tipini değişken ismini

```
public int En;
public int Boy;
```

- *Bu özellikler kullanılarak **alan** hesaplayan bir metot oluşturulabilir.*

- ***Metot tanımlama kalıbı;***

önce,	*sonra*	*daha sonra*	*en son*
***erişim türünü**(`public`)*	***geri döndürülen değer tipini**(`int`),*	***metot ismini** (AlanHesapla),*	***parametre-argümanlar***

erişim türü geri döndürme tipi metot ismini (parametre)

En ve Boy özellikleri kullanılarak iki adet metot oluşturalım

- *En ve boy değerlerinin çarpımı sonucu alanı hesaplayalım*

```
public int AlanHesapla()
    {
        return En * Boy;
    }
```

- *Birde **çevre** hesaplayan bir metot oluşturulabilir.*

```
public int CevreHesapla()
    {
        return 2 * En + 2 * Boy;
    }
```

- *Birde BilgiYaz diye metot oluşturalım;*

Dortgenin En ve Boy değerlerini ekrana yazdıralım.

```
public void BilgiYaz()
        {
            Console.WriteLine(En);
            Console.WriteLine(Boy);
        }
```

- *Bir burada dörtgen isminde bir class oluşturduk bunu altında da "**AlanHesapla**" ve "**CevreHesapla**" ve "**BilgiYaz**" diye 3 adet metot tanımladık.*
- *Bu sınıf sayesinde kenar uzunlukları farklı değerde olan bir çok dörtgen tanımlanabilir.*

Dortgen.cs* | Program.cs*

Ornek17._0SınıfTanımlama.Dortgen

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Ornek17._0SınıfTanımlama
{
    public class Dortgen
    {
        public int En;
        public int Boy;

        public int AlanHesapla()
        {
            return En * Boy;
        }

        public int CevreHesapla()
        {
            return 2 * En + 2 * Boy;
        }

        public void BilgiYaz()
        {
            Console.WriteLine(En);
            Console.WriteLine(Boy);
```

- ***Şimdi tanımlamış olduğumuz bu sınıfı ana programımıza dönüp nasıl kullanacağımıza bakacak olursak;***

- *Ana programımızda daha önceden tanımladığımız bir sınıfı kullanmak istediğimizde önce o sınıftan bir nesne türetmemiz gerekiyor.*

- *Nesne türetmek için ilk kullanılırken biz buna " **inşa etmek**" diyoruz. **Yapıcı bir metot** tanımlıyoruz. new anahtar kelimesi ile bu tanımlamayı yaptığımızda bu sayede bize bellekten de yer ayırılmaktadır. Bu dörtgen sınıfı içerisindeki tüm özellikler ve tanımlar için bellekten yer ayrılır. Bu yapıcı metotlara sonra bakalım.*

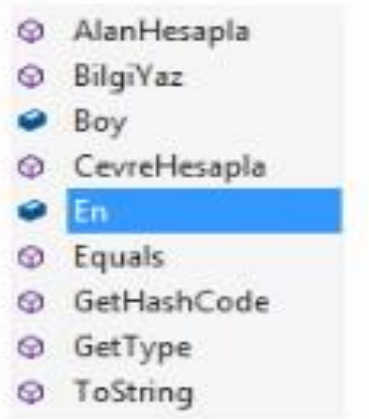
- *İki adet nesne oluşturalım;*

```
Dortgen nesne1 = new Dortgen();
Dortgen nesne2 = new Dortgen();
```

Ana program;

```
Dortgen nesne1 = new Dortgen();
Dortgen nesne2 = new Dortgen();

nesne2.
```



- *Bunlar birbirinden farklı iki adet nesnedir.*
- *Nesne 2 ye en ve boy değerleri atayalım;*

Bundan sonra ;

nesne2 yazıp noktaya bastığımızda;

- *sınıf içerisinde tanımladığımız değişken ve metotların listesini getirir.*

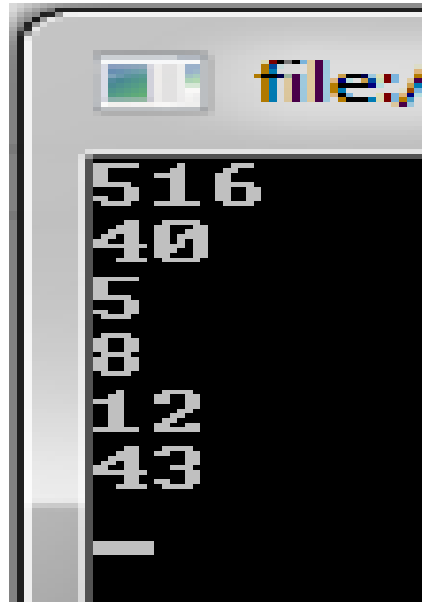
```
nesne2.Boy = 8;
nesne2.En = 5;
nesne1.Boy = 12;
nesne1.En = 43;
```

Ana program;

- *Nesnemize ait metotları da çağırabiliriz. Mesela alan hesaplayan metodu çağırıp bunu ekrana yazdıralım;*

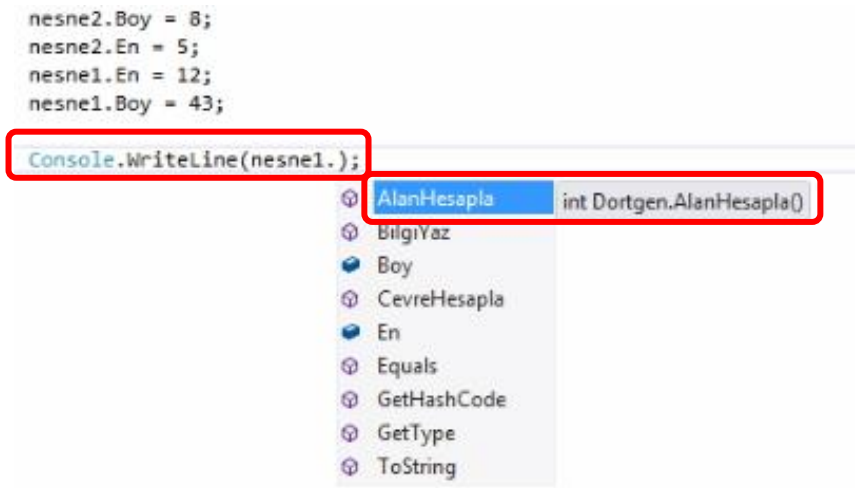
```
Console.WriteLine(nesne1.AlanHesapla());
Console.WriteLine(nesne2.AlanHesapla());
nesne2.BilgiYaz();
nesne1.BilgiYaz();
Console.ReadLine();
```

Çalıştırıp bakalım;



Ana program;

- *Burada AlanHesapla metodunun üzerinde beklediğimizde int türünden geri değer döndürdüğünü her hangi bir argüman istemediğini görebiliriz. Visual Studio bize bu konuda yardımcı olmaktadır.*
- *Aslından burada kullandığımız her nesne sınıf kütüphanesinden alarak oluşturuyoruz.*
- *Şöyle ki bir programcının tanımladığı bir sınıf nesne diğer bir programcı kullanarak kendi programını yazar.*
- *Kullanılacak olan metot değer ve veri dönüş değerleri hakkında bilgi verir*



Ana program;

```
namespace Ornek17._0SınıfTanımlama
{
    class Program
    {
        static void Main(string[] args)
        {
            Dortgen nesne1 = new Dortgen();
            Dortgen nesne2 = new Dortgen();

            nesne2.Boy = 8;
            nesne2.En = 5;
            nesne1.En = 12;
            nesne1.Boy = 43;

            Console.WriteLine(nesne1.AlanHesapla());
            Console.WriteLine(nesne2.AlanHesapla());

            nesne2.BilgiYaz();
            nesne1.BilgiYaz();

            Console.ReadLine();

        }
    }
}
```

Sınıf;

```
using System;

namespace Ornek17._0SınıfTanımlama
{
    public class Dortgen
    {
        public int En;
        public int Boy;

        /// <summary>
        /// Bu metot en ve boy niteliklerine bakraka nensnenin alanını hesaplar. Parametre almadan çalışır
        /// </summary>
        /// <returns>Bu metot tam sayı olarak hesaplanan alan değerinni döndürür</returns>
        public int AlanHesapla()
        {
            return En * Boy;
        }
        ///
        public int CevreHesapla()
        {
            return 2 * En + 2 * Boy;
        }

        public void BilgiYaz()
        {
            Console.WriteLine(En);
            Console.WriteLine(Boy);
        }
    }
}
```

Ana program;

- *Metot ile ilgili açıklama;*
- *ilgili metodun önüne gelip 3 adet /// ekleyerek* ***Summary*** *blokları oluşturabilir.*
- ***Summary*** *arasına* ***metot ile ilgili açıklama,***
- ***returns*** *arasına* ***geri döndürdüğü değer*** *ile ilgili açıklama yazılır.*