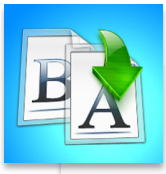
NESNE YAPISINA GİRİŞ



İÇİNDEKİLER

- Nesne Yönelimli Programlama
  - Sınıf
    - Sınıf Özellikleri
    - Sınıf Metotları
  - Nesne
  - Kalıtım
  - Çok Biçimlilik
  - Kapsülleme

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Nesne yönelimli programlamanın tanımını yapabilecek,
  - Sınıf kavramını anlayarak projelerde sınıflar oluşturabilecek,
  - Oluşturduğu sınıflardan nesneler türetip, nesne özelliklerine değerler atayabilecek,
  - Kalıtım, çok biçimlilik ve kapsülleme kavramlarının nasıl kullanılabileceğini anlayabileceksiniz.

Atatürk Üniversitesi
Açıköğretim Fakültesi

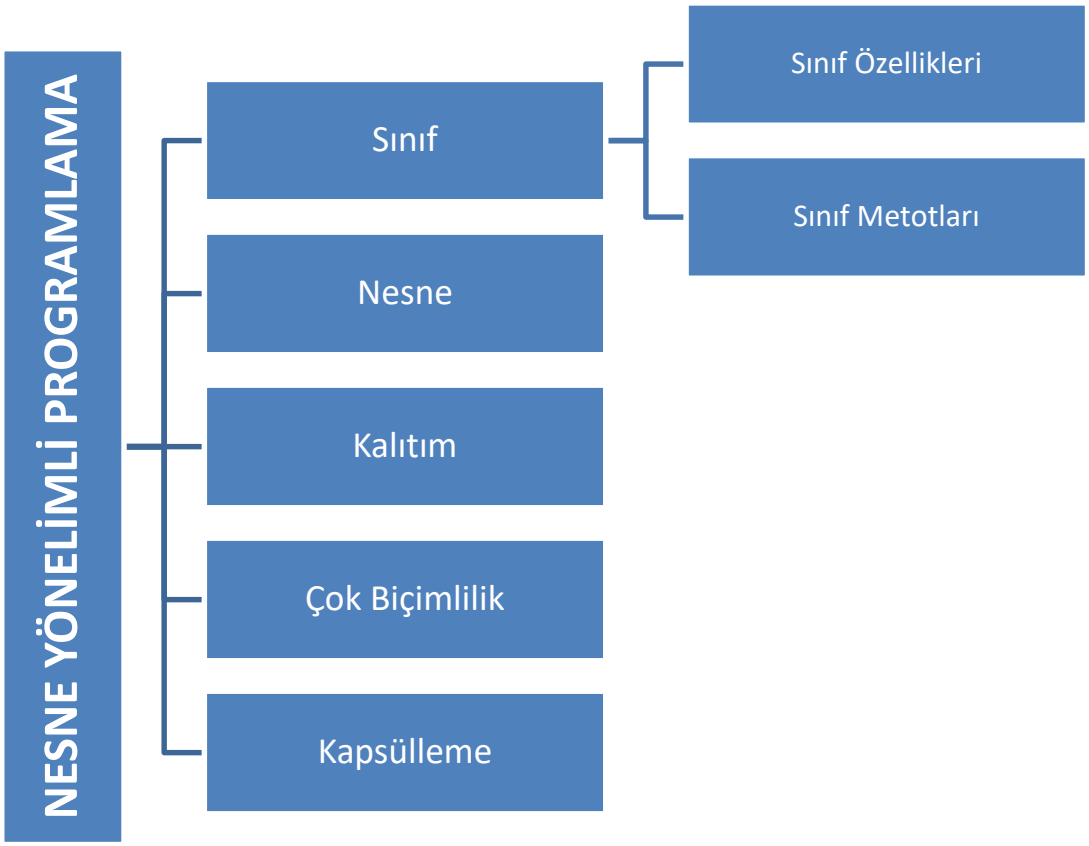
NESNE TABANLI PROGRAMLAMA I

Öğr. Gör.
Daha ORHAN

ÜNİTE

4

© Bu ünitenin tüm yayın hakları Atatürk Üniversitesi Açıköğretim Fakültesi'ne aittir. Yazılı izin alınmadan ünitenin tümünün veya bir kısmının elektronik, mekanik ya da fotokopi yoluyla basımı, yayımı, çoğaltımı ve dağıtımı yapılamaz.


NESNE YÖNELİMLİ PROGRAMLAMA
Sınıf
Sınıf Özellikleri
Sınıf Metotları
Nesne
Kalıtım
Çok Biçimlilik
Kapsülleme

GİRİŞ

Günümüz dünyasında yaptığımız işlemleri kolaylaştırmak adına her geçen gün donanım birimlerinin, yazılımlarla desteklenerek kullanımı yaygınlaşmaktadır. Yazılımlar sadece bilgisayarlar üzerinde gerçekleştirilmemektedir. Etrafımızda gördüğümüz çamaşır makinesi, buzdolabı, uydu alıcı, yazıcı, tablet gibi birçok cihazın içinde yazılım geliştiriciler tarafından hazırlanan ve belli bir amaca hizmet eden kod parçacıkları bulunmaktadır. Yazılım geliştiricilerin hazırladığı bu kod parçacıklarına programlama, donanım birimlerinin programlama ile bir işi gerçekleştirmesine yönelik oluşturulan makine komutlarına da yazılım denmektedir.

.Net Framework kurulu olan herhangi bir bilgisayarda C# uygulamaları geliştirebilirsiniz.

Yazılımlar geliştirilirken tercihe bağlı olarak birden çok programlama dili içinden herhangi birini seçebilirsiniz. Bu ünitede C dilinden türeyen nesne yönelimli programlama (OOP) yöntemini kullanan C# dili kullanılmaktadır. Ayrıca program yazarken karşılaşılan karmaşıklıkların giderilmesi temeline dayanarak geliştirilen nesne yönelimli programlama önemini ortaya koyan nesne kavramı tanıtılacaktır.

.Net Framework kurulu olan herhangi bir bilgisayarda C# uygulamaları geliştirebilirsiniz. Geliştirdiğiniz uygulamalar not defteri üzerine yazılıp DOS ortamında derleniyorsa biraz zahmetli olabilir. Bu yüzden Microsoft firmasının 1997 yılından beri geliştirmeye başladığı Visual Studio .Net geliştirme ortamından istifade etmek faydalı olacaktır. Bu ünitede, farklı programlama dilleri kullanmanıza da izin veren ve masaüstü, mobil, web sitesi gibi uygulamalar geliştirmeye imkan tanıyan Visual Studio .Net 2019 geliştirme ortamı kullanılmaktadır.

Ayrıca bu ünitede tanıtılan kavramlar, nesne yönelimli programlamayı (OOP) anlamak için zemin hazırlamaktadır. Diğer konulara geçmeden önce burada tanıtılan kavramlar konusunda kendinizi rahat hissetmeniz gerekmektedir. Ünite tamamlandıktan sonra nesnelerin, özelliklerin ve yöntemlerin ne olduğunu bilmenizle beraber bunların bir nesne yönelimli programlamada nasıl kullanıldığını da bilmeniz gerekmektedir. Eğer ünitede tanıtılan kavramların zihninizde tam olarak yer etmediğini düşünürseniz bu üniteyi tekrar gözden geçirmeniz tavsiye edilmektedir.

NESNE YÖNELİMLİ PROGRAMLAMA

Nesneleri kullanarak gerçekleştirilen programlama anlayışı uzun yıllardır var olsa da, nesneler son zamanlarda programlamanın vazgeçilmez bir öğesi haline gelmiştir. Günümüzde sıklıkla kullanılan nesne yönelimli programlama anlayışından bahsedebilmek için programlama dilinde aramamız gereken bir takım kavramlar bulunmaktadır. Bu kavramlar;

- Sınıf,
- Nesne,
- Kalıtım,

- Çok biçimlilik,
- Kapsüllemedir.

Birbiriyle bağlantılı olan bu kavramlardan Nesne kavramını ele almak için diğer kavramlara da detaya inilmeden değinilecektir. Özellikle sınıf ve nesne kavramları içi içe geçmiş bir şekilde karşımıza çıkmaktadır. Soyut bir halde bulunan bu kavramlar günlük hayattan örnekler verilerek anlatılmaya çalışılacaktır.

Sınıf kavramını bir şablon olarak düşünebiliriz. Örneğin bir yemek hazırlamak için tariflerin bulunduğu kitaptaki bir sayfada yemek tarifini sınıf olarak düşünebiliriz. İlgili sayfadaki yemek tarifine bağlı olarak ele alınan malzemelerin kullanımı ile ocak üstünde pişirilen ürün ise bizim nesnemizdir. Yani bu durumu bilgisayar ortamında ele aldığımız zaman bir sınıfın RAM üzerinde kapladığı alan nesnedir.

Nesneler sınıfların birer üyesidir.

Sınıf

Sınıf kavramını günlük hayatımızla ilişkilendirecek olursak, gerçek dünyada gözlemlediğiniz bazı nesnelerin soyut olarak ifade edilmesi veya basitleştirilmesi olarak tanımlayabiliriz. Nesneler ise bu sınıfların birer üyesidir. Bir sınıfı temel olarak iki bileşene ayırmamız mümkündür. Bunlar;

- Nesneyi tanımlayan özellikler,
- Nesneyle ilişkilendirmek istediğimiz yöntemler veya eylemlerdir.

Bu ünitede sınıf ve nesne kavramı anlatılırken günlük hayattan bir örnek ele alınacak olup ardından Visual Studio .Net ortamında uygulaması gerçekleştirilecektir. Bu doğrultuda basit bir araba sınıfı yapalım ve bu sınıfın üyelerini belirleyelim.

- Araba
  - Modeli
  - Kasası
  - Uzunluğu
- Yakıtı

Yukarda araba sınıfına ait bir şablon oluşturulmuştur. Bu şablonda bulunan her bir özellik araba sınıfını göstermektedir. Şimdi bu sınıfı daha somut bir hale getirelim. Yani Araba sınıfına ait nesnemizi oluşturalım.

- Mercedes
  - S320
  - Sedan
  - 5 metre
  - Dizel

Böylelikle soyut olan bir şablon yerine Araba sınıfından türemiş bir *Mercedes* nesnesi elimizde bulunmaktadır. Böylelikle araba artık bilgisayar belleğinde yer kaplamaktadır. Araba sınıfında bulunan özellikler Nesne ile eşleştirilmektedir. Bu özelliklerden herhangi birinin değişmesi durumunda

nesnemizin de değişeceğini unutmamamız gerekir. Yani araba sınıfı içinde tanımlanan özelliklerde yer alan bilgileri değiştirerek nesneyi değiştirebilirsiniz.

Sınıf özellikleri

Bir nesneyle ilişkilendirmek ve kaydetmek istediğiniz veriler, sınıfın özelliklerini oluşturmaktadır. Sınıf özelliklerinin değerlerini değiştirerek nesnenin durumunu değiştirebilirsiniz. *Bir nesnenin durumu, nesneyi tanımlamak için kullanılan özelliklerin değerleriyle belirlenir.* Araba sınıfına ait özellikler aşağıdaki gibidir;

- Modeli
- Kasası
- Uzunluğu
- Yakıtı

Bu özellik değerlerinden herhangi biri değişirse, nesnenin durumu da değişir. Yani yeni bir nesne ortaya konulmuş olacak ve bellekte o sınıfa ait farklı bir nesne yer alacaktır. Bir sınıfa ait herhangi bir özelliğin değeri her değiştiğinde, nesnenin durumu da değişmektedir.

Sınıf metotları, nesnenin gerçekleştirebileceği davranışları veya eylemleri belirlemektedir.

Sınıf metotları

Bir nesneye ait durumu tanımlamak için özellik değerleri kullanılmaktadır. Sınıf metotları ise bu özelliklere etki etmektedir. Buradan hareketle metot kavramını, *nesnelerin belirli işlemlerini gerçekleştirmeyi sağlayan fonksiyonlar* olarak tanımlayabiliriz. Kısaca sınıf metotları, nesnenin gerçekleştirebileceği davranışları veya eylemleri belirlemektedir.

Metotlar genel olarak değer (parametre) alabilen, değer döndüren ve birden fazla değer ile işlem yapabilen yapıdadırlar. Metotları nesnelerle ilişkilendirmek istediğimiz eylemleri tanımlamak için kullanırız. Araba sınıfında verdiğimiz örneği dikkate alacak olursak metotlarını aşağıdaki gibi tanımlayabiliriz;

- Çalıştırma
- Durdurma
- Hızlanma

Böylelikle araba sınıfında bulunan nesnelerimizi istediğimiz zaman çalıştırabilir, hızlandırabilir veya durdurabiliriz. Araba sınıfına ait Model, Kasa Uzunluk ve Yakıt olmak üzere 4 özellik ve Çalıştırma, Durdurma ve Hızlanma olmak üzere 3 metot tanımlanmıştır. Bu özellik ve metotların sayısı istenildiği kadar artırılabilir. Ancak burada önemli olan, bu özellik ve metotların sade ve minimum sayıda olmasıdır. Çünkü programlamanın temel mantığı çözüme gidecek en hızlı ve en sade yöntemi belirlemektedir. Bununla birlikte nesne yönelimli programlamanın önemli avantajlarından biri olan “kodun yeniden kullanımı” durumuna da uygunluk göstermiş olursunuz.

Günlük hayatta yaşam boyu yaptığımız planlamayı bilgisayar ortamında da gerçekleştirmemiz gerekiyor. Bu yapılan planlama işlemine bilgisayar ortamında ise *algoritma* denilmektedir. Projelerimize başlamadan önce algoritmalarımızı

hazırlamak yapacağımız projelerin daha sağlıklı bir şekilde yürütülmesine imkan tanıyacaktır. Bu yüzden kısaca program yazma adımlarına da değinmenin faydalı olacağı düşünülmektedir. Bu adımlar;

- İş veya problem irdelenerek netleştirilmelidir.
- Sonuca en az komut kullanılarak, en kısa sürede ulaşılması sağlanacak şekilde ve sonucu kusursuz olarak elde etmek için en hassas çözüm yolu seçilmeli veya belirlenmelidir.
- Programın akış diyagramı çizilmeli veya algoritması yazılmalıdır.
- Hazırlanan algoritma veya çizilen akış diyagramı programcının kullanabildiği bir programlama diliyle program oluşturulmalıdır.

Bir sınıfta ne kadar özellik ve metot varsa, o özellik ve metotları kullanmak için o kadar kod yazmanız gerekmektedir. Bu yüzden sadeliği ön planda tutmanız ve sınıfınızın planlamasını hazırlıklı bir şekilde gerçekleştirmeniz gerekiyor. Bir sınıfa ait kodlar ne kadar genel olursa o kodların başka bir projede kullanımı da o kadar kolay olmaktadır. Bununla birlikte özellik ve metotlar bir nesnenin ihtiyaçlarını karşılayabilecek düzeyde ayarlanmalı ve projelerin ana amacı gözden kaçırılmamalıdır. Yani bir araba sınıfı için araç kullanıcılarının arabada arayacağı özellikler ile çevreye olan zararlarını gözlemlemek isteyen bir kurumun arayacağı özellikler arasında farklılıklar olabilir. Bu nedenle bir sınıf tasarladığınızda, yazdığınız kodu en aza indirmek ile sınıf için tasarım hedeflerini yerine getirmek arasında bir denge kurmanız gerekir.

Hazırladığımız programlarda gerçekten kullanabileceğimiz yapılar nesnelerdir.

Nesne

Bir sınıfa ait özellikler nesneler sayesinde kullanılmaktadır. Nesneler sınıflardan türetilirler. Tek başlarına bir anlam ifade etmeyen bu yapılar bir sınıfa ihtiyaç duymaktadırlar. Daha öncede belirttiğimiz gibi nesne ve sınıf kavramları birbiriyle çok ilişkilidir. Bir nesne aynı zamanda bir sınıfın örneği olarak da adlandırılır. Yani örnekleme, bir nesneyi tanımlamak için bir sınıfı kullanma eylemidir. Hazırladığımız programlarda gerçekten kullanabileceğimiz yapılar nesnelerdir. Bir sınıftan nesne türetme işlemi aşağıda gösterildiği gibi gerçekleşmektedir.

```
Sınıf_adı Nesne_adı = New Sınıf_adı();
```

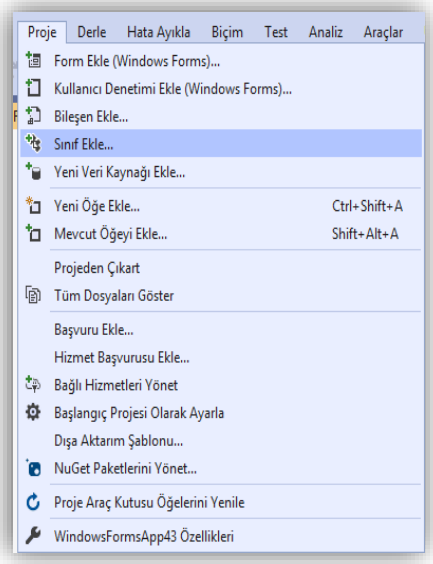
Öncelikle nesnenin türetileceği sınıf adı yazılır ve bir boşluk bırakıldıktan sonra türetilecek olan nesnenin adı yazılmaktadır. Ardından eşittir sembolü kullanılarak *New* anahtar kelimesiyle birlikte sınıfın adı tekrardan yazılır. Böylelikle belirtilen sınıfa ait bir nesne türetilmiş olur. Burada kullanılan New anahtar kelimesi ile sınıftan nesne türetilmiştir. Burada kullanılan metoda *Yapıcı Metot* ve *Constructor* denilmektedir. Bu metotların detaylarına ilerleyen ünitelerde değinileceği için şimdilik bu kadar bilgi sahibi olmanız yeterli görülmektedir.

Nesne türetme işlemi gerçekleştirilirken nesne adını belirlediğimiz alanda belirli bir yazım tarzı benimsememiz hazırladığınız kodların okunabilirliğini olumlu yönde etkileyecektir. Bununla beraber isimlendirme kurallarını da dikkate almanız

gerekmektedir. İsimlendirme kurallarını ve dikkat edilmesi gereken kriterleri kısaca hatırlatmanın faydalı olacağı düşünülmektedir. İsimlendirme kuralları;

- İsimlendirmeye sayı ile başlanılmamalıdır.
- Boşluk ve özel karakterler (?,],!,…) kullanılmamalıdır.
- Programlama dilinde var olan (for, and, not,vb.) anahtar kelimeler kullanılmamalıdır.
- Türkçe karakterler (ç,ş,ğ, vb.) kullanılmamalıdır.

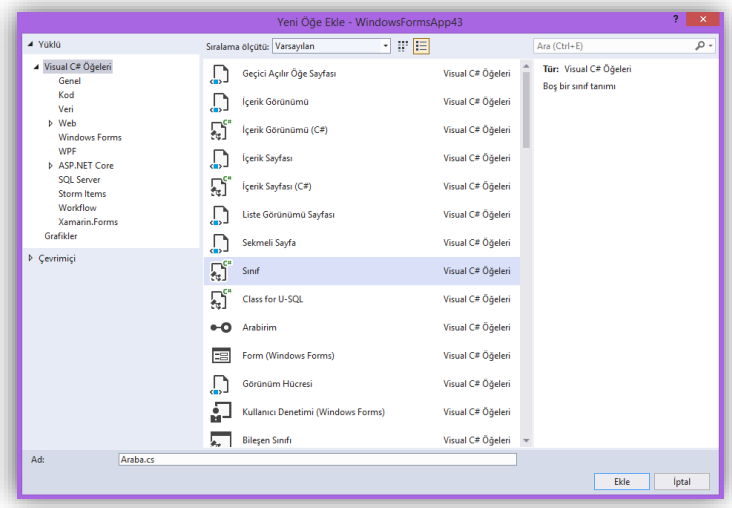
Sınıf kavramını anlatırken örneklendirdiğimiz Araba sınıfı ve o sınıfa ait özellikleri belirleyeceğimiz somut bir uygulama geliştirelim. Bu uygulamada sınıfın özellikleri türetilen nesne yardımıyla kullanılabilir hale getirilecektir. Öncelikle Visual Studio .Net 2019 geliştirme ortamını çalıştırıp C# programlama dilini seçerek *Windows Forms Uygulaması* başlatalım. Bu işlemin ardından Şekil 4.1.'de gösterildiği gibi menü çubuğunda bulunan *Proje* menüsündeki *Sınıf Ekle…* seçeneğini işaretleyelim.



Şekil 4 1. Sınıf Ekle Menü Yolu

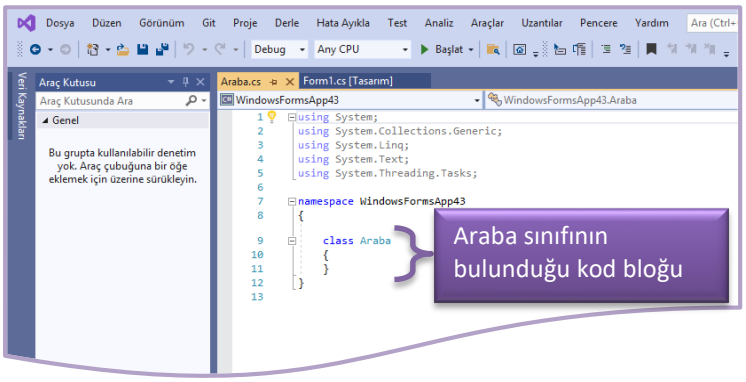
Projemize ilk defa sınıf ekliyorsak bu değer varsayılan olarak Class1 değerini almaktadır.

Menü yolu kullanıldıktan sonra *Yeni Öğe Ekle* penceresi açılacaktır. Bu pencerede, Şekil 4.2.'de gösterildiği gibi sınıf seçeneği işaretlenip Ad alanında bulunan uzantıdan önceki kısım doldurulmalıdır. Eğer projemize ilk defa sınıf ekliyorsak bu değer varsayılan olarak Class1 değerini almaktadır. Uygulamamızda bu alanı *Araba* olarak değiştirip Ekle butonuna tıklıyoruz. Burada .cs uzantısını değiştirmemeli veya silmemelisiniz.



Şekil 4. 2. Yeni öğe ekle penceresi

Yeni öğe ekle penceresinde gerekli düzenlemeler yapılıp ekle butonuna tıklandıktan sonra karşımıza Şekil 4.3.'de görmüş olduğumuz kod editörü gelecektir. Bu işlem ile projemizde Araba sınıfını oluşturmuş olduk.



Şekil 4. 3. Kod editöründe araba sınıf

Şimdi Araba sınıfından türeteceğimiz nesnelerle ilişkilendirmek istediğimiz özellikleri belirlememiz gerekiyor. Sınıf özelliklerini anlattığımız başlıkta bu özellikleri model, kasa, uzunluk ve yakıt olarak belirlemiştik. Bu özellikleri kod editöründe tanımlamak için aşağıdaki kodları yazmamız gerekmektedir.

Projemize ilk defa sınıf ekliyorsak bu değer varsayılan olarak Class1 değerini almaktadır.

```
class Araba
  {
        public string model;
        public string kasa;
        public double uzunluk;
        public string yakit;
  }
```

Örneğimizde olduğu gibi kod editöründe de 4 adet değişkenimizi tanımladık. Bu değişkenler Public erişim belirleyicisi kullanılarak tanımlanmıştır. Erişim

belirleyicilere ilerleyen ünitelerde yer verilecek olsa da kısaca burada kullanım amacını belirtmenin faydalı olduğu düşünülmektedir. Burada erişim belirleyicimizi belirtmediğimiz takdirde tanımladığımız değişkenler özel (private) olarak kabul edilmekte ve bu değişkenlere erişim sağlanamamaktadır. Projemizde var olan diğer formlardan da bu değişkenlere erişim sağlayabilmek için Public erişim belirleyicisi kullanılmıştır.

Değişkenlerimizden 3 tanesi *string* türünde, bir tanesi ise *double* türündedir. Değişkenler tanımlanırken tutacağı değerlere göre bu veri türlerinin doğru bir şekilde ayarlanması gerekmektedir. Aksi takdirde projenin çalışma anında hatalı sonuçlar ile karşılaşılabilmektedir. Bu işlemler tamamlandıktan sonra sınıfımızdaki özelliklere işlevsellik kazandırmak için nesnemizi türetmemiz gerekmektedir. Bunun için yazmamız gereken kod satırı aşağıda verilmiştir;

- Araba mercedes = new Araba();

Bu kodların nasıl çalıştığını gözlemleyebilmek için projemizin tasarım alanına geçelim. Tasarım alanında bulunan formumuz üzerine araç kutusundan ListBox kontrolünü sürükle bırak yöntemi ile yerleştirelim. Bu işlemin ardından form üzerinde çift tıklayarak kod editöründe forma ait açılan yüklenme olayına aşağıdaki kodları yazalım.

```
private void Form1_Load(object sender, EventArgs e)
{
        Araba mercedes = new Araba();
        mercedes.model = "s320";
        mercedes.kasa = "sedan";
        mercedes.uzunluk=5.8;
        mercedes.yakit = "dizel";
        listBox1.Items.Add(mercedes.model);
        listBox1.Items.Add(mercedes.kasa);
        listBox1.Items.Add(mercedes.uzunluk);
        listBox1.Items.Add(mercedes.yakit);
}
```

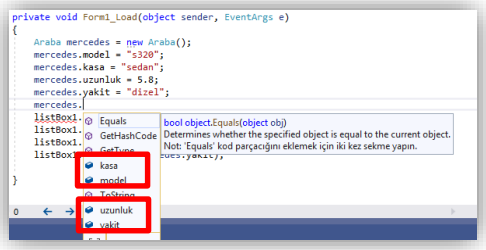
Bir nesnenin türetildiği sınıfta tanımlanan özellik ve metodları görüntülemek için nesne adını yazdıktan sonra nokta (.) konulmalıdır.

Bir nesnenin durumunu değiştirmekle ilişkili yazım kurallarına dikkat etmek önemlidir. Bir programlama dilinin yazım kuralları, dilin kullanımını yöneten kuralları ifade etmektedir. C# programlama dilinde bir nesnenin özelliğini kullanmak için genel sözdizimi aşağıdaki gibidir:

Nesne_adı.Özellik

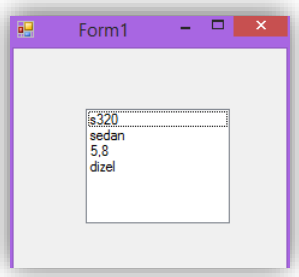
Bu sözdizimi kullanıldıktan sonra *Nesne_adı* nesnesi için bellekte bir yer ayrılır. Artık Nesne_adı isimli nesne kullanılarak türetildiği sınıfta tanımlanan özellik ve metotlara erişilebilir. Türetildiği sınıfta tanımlanan özellik ve metodları görüntülemek için Nesne_adı yazdıktan sonra nokta (.) konulmalıdır.

Yukarda verilen kodları yazarken kopyala yapıştır yöntemini kullanmamanız tavsiye edilmektedir. Çünkü bir programlama dilinin yazım kurallarını öğrenmek için pratik yapmanın ve sunmuş olduğu özelliklerin gözlenmesi tecrübe kazandıracaktır. Kodları yazarken araba sınıfında tanımladığınız özelliklerin, nesne adı yazılıp nokta konulduktan sonra özellikler listesinde de yer aldığını göreceksiniz. Yapmış olduğumuz uygulamada bu durum, Şekil 4.4.'de gösterildiği gibidir.



Şekil 4.4. Özellikler listesi (intellisense)

Kod yazım kurallarını dikkate aldığımızda *eşittir (=) operatörünün* aktarma işleminde kullanıldığı da dikkatimizi çekmektedir. Bu operatör yardımıyla ifadenin sağ tarafında bulunan değer sol tarafta bulunan özellik içine aktarılmaktadır. Böylece bu özellikler kullanıcıya istenilen bir yöntemle iletilmektedir. Bu uygulamada listbox kontrolünden istifade edilmiştir. Belirtilen işlemler tamamlandıktan sonra uygulamayı çalıştırdığımız zaman Şekil 4.5.'deki ekran görüntüsü elde edilecektir.



Şekil 4.5. Çalıştırılan uygulamanın ekran görüntüsü

Nesne yönelimli programlamanın avantajları arasında bulunan "kod tekrarını azaltma" hususuna değinilmişti. Sınıf yapısı bu anlamda çok önemli bir şekilde bize yardımcı olmaktadır. Bu yüzden yapmış olduğumuz uygulamayı biraz daha genişletip Araba sınıfına ek olarak *Fiyat* sınıfı oluşturup bunların özelliklerini tek bir form üzerinde kullanalım. Sınıf oluşturma işlemi ve özelliklerini belirleme yukardaki örnekte anlatıldığı için tekrar değinilmeyecektir. Bu doğrultuda kodlarımızı aşağıdaki gibi düzenleyelim.

Eşittir (=) operatörü aktarma işleminde kullanılabilmektedir.

```
class fiyat
    {
        public string tl;
```

```
    }
    private void button1_Click(object sender, EventArgs e)
    {
        Araba mercedes = new Araba();
        mercedes.model = "s320";
        mercedes.kasa = "sedan";
        mercedes.uzunluk = 5.8;
        mercedes.yakit = "dizel";
        Fiyat arabafiyat = new Fiyat();
        arabafiyat.tl = Interaction.InputBox("araba fiyatını giriniz");
        MessageBox.Show(mercedes.model+ " model " + mercedes.kasa+
" kasa ... arabanın fiyatı "+arabafiyat.tl+ " tl'dir");
    }
```

Tek sınıf kullanarak yaptığımız uygulamadan farklı olarak, bu uygulamada kodlar formun yüklenme olayına değil form üzerine eklenen bir buton kontrolünün tıklanma olayına yazılmıştır. Bununla beraber elde edilen sonuç mesaj penceresinde gösterilmiştir. Ayrıca arabanın fiyatı kullanıcıdan inputbox penceresi yardımıyla string türünde alınmıştır. C# uygulamalarında inputbox diyalog penceresini kullanmak için başvuru eklemek gerekmektedir. Bu işlem bu ünitenin konusu olmadığı için aşağıda verilen bireysel etkinliği yapmanız tavsiye edilmektedir. Böylelikle konuyu kavrama noktasında daha ileri seviyede olacağınız düşünülmektedir.

Bireysel Etkinlik

- Araba ve Fiyat isimli iki adet sınıf oluşturup bu sınıfın özelliklerini farklı veri türlerinde oluşturunuz. Farklı veri türlerinin değerlerini, form üzerine yerleştirdiğiniz textbox kontrollerinden alıp buton yardımıyla listbox kontrolünde görüntüleyiniz. (NOT: tür dönüşümleri kullanmanız gerekebilir!)

Kalıtımdan bahsedebilmek için bir sınıf kendinden üstte bulunan sınıfın özelliklerini taşıması gerekmektedir.

Kalıtım

Kalıtımdan bahsedebilmek için bir sınıf kendinden üstte bulunan sınıfın özelliklerini taşıması gerekmektedir. Böylelikle bir sınıfa ait özelliklerin bir veya daha fazlasını başka sınıf veya sınıflar içinde kullanabiliriz. Projelerimizde kalıtımı kullanabilmek için *iki nokta üst üste (:)* kullanmamız gerekmektedir. Bu işlem ":" operatöründen önce sınıfımızın ismini, sonra da kalıtım alınacak sınıfın ismini yazarak gerçekleştirilmektedir.

Kalıtım kavramını anlayabilmek için araba sınıfımızın üstünde araç sınıfı olduğunu varsayalım. Araç sınıfının özelliklerini aşağıda gibi sıralayalım;

- Motor
- Direksiyon
- Teker

Araba sınıfımızın özelliklerini tanımlamıştık. Ek olarak otobüs sınıfı tanımlayalım. Bu otobüsü sınıfa ait özellikler aşağıda verilmiştir;

- Yolcu taşıma kapasitesi
- Koltuk düzeni
- Kat sayısı

Bu özellikler göz önünde bulundurulduğunda araç sınıfına ait özelliklerin kalıtım ile araba ve otobüs sınıfında alındığını görebiliyoruz. Ayrıca Araba ve otobüs sınıflarının kalıtımla gelen özelliklerin haricinde kendine ait özelliklerinin olduğu da anlaşılabiliyor. Bu durum araba ve otobüs sınıfın durumunu değiştirmediğini göstermektedir. Yani bu sınıflarında kalıtım yoluyla alt sınıfları barındırması mümkündür. Bu sınıfların içinde nesneler türetilmediği müddetçe bu özelliklerin kullanılamayacağı unutulmamalıdır.

Otobüs sınıfı altında bir nesne oluşturup özelliklerini inceleyelim. Nesnemizin adı Tourismo olsun. Bu nesne kalıtımla 5000 motor, 1 direksiyon ve 12 teker sayısına sahip özellik değerlerini almaktadır. Bununla beraber kendine ait özellikleri de 60 yolcu kapasiteli, 2+1 koltuk düzeni ve 2 katlı özelliklerini taşımaktadır.

Çok Biçimlilik

Kalıtımla bir sınıftan alınan özellik olduğu gibi kullanılıyorken, çok biçimlilik işleminde bu özellik farklı şekillerde de kullanılabilmektedir. Bu işleme aynı zamanda *Polimorfizm* de denilmektedir. Çok biçimlilik durumunu örneklemeden önce birkaç teknik terimden kısaca bahsedilecektir.

Virtual: Bir sınıftan alınan kalıtımın değiştirilebilmesi için kullanılan komut.

Override: Bu komut ise kalıtım ile gelen metodun bulunduğu sınıftaki değer yerine başka bir değer kullanılarak işletilmesini sağlamaktadır.

Kalıtımla bir sınıftan alınan özellik olduğu gibi kullanılıyorken, çok biçimlilik işleminde bu özellik farklı şekillerde de kullanılabilmektedir.

Komutlar ve tanımlarından da anlaşılacağı üzere çok biçimlilik, kalıtımla gelen metotların farklı biçimlerde kullanılmasına imkan tanıyan bir özelliktir. Şimdi araç sınıfımızı tekrar ele alalım. Araç sınıfına ait özellik ve metotlar aşağıda belirtilmiştir.

Araç sınıfının özellikleri;

- Motor
- Direksiyon
- Teker

Araç sınıfının metodu;

- Önden çekiş (çok biçimli)

Şimdi araç sınıfının altında bulunan araba sınıfına ait kalıtımla gelen özellik ve metotlarını inceleyelim. Özellikler olduğu gibi aktarılırken metodunun farklı bir biçimde olduğunu göreceksiniz. Ayrıca özellikler ve metot Mercedes nesnesi üzerinden gösterilmiştir.

Araba sınıfına ait Mercedes nesnesinin özellikleri;

- Motor = 3200;
- Direksiyon = Fonksiyonel;
- Teker = 4 tekerli;
- Model = “S320”;
- Kasa = “Sedan”;

Araba sınıfına ait Mercedes nesnesinin metodu;

- Dört tekerden çekiş desteği

Anlaşıldığı üzere bir metot önden çekiş, 4 teker veya arkadan itiş şeklinde çekiş sistemi dikkate alındığı zaman farklı biçimlerde kullanılabilmektedir. İşte bu durum çok biçimlilik olarak karşımıza çıkmaktadır.

Kapsülleme

Kapsülleme nesne yönelimli programlamanın önemli özelliklerinden birisidir. Bazı durumlarda nesnelerin özelliklerini gizlemek zorunda olabiliriz. Yani daha öncede bahsedildiği gibi nesne adı yazıldıktan sonra nokta operatörü konularak açılan özellikler listesini kısıtlamak isteyebilirsiniz. Biraz daha açacak olursak, verileri programın diğer bölümleri tarafından yanlışlıkla yapılan değişikliklerden korumak için bir nesnenin içine yerleştirirsiniz. Bu şekilde yapılan bir nesne içindeki verileri gizleme işlemine kapsülleme denilmektedir.

Erişim denetleyicilerle de yakından ilgisi olan kapsülleme işlemi ilerde daha detaylı anlatılacak olsa da burada bazı özelliklerinden bahsetmenin faydalı olacağı düşünülmektedir. Özellikle nesnelerin yanlışlıkla özelliklerinin değiştirilmesini engellemek ve arka planda çalışan kodlarını gizlemek amacıyla kullanılan kapsülleme işleminin iki önemli kullanımı mevcuttur. Bunlar;

- Sınıflandırmayı desteklemek ve
- Sınıf kullanımını denetlemektir,

Kapsülleme etkileşimiyle değişkenlerimizi istediğimiz formatta kullanmak için *private* olarak tanımlarız. Ayrıca Kapsülleme işleminde bilmemiz gereken 3 önemli kavram bulunmaktadır. Bunlar;

Get Bloğu: Geriye gönderdiğimiz değerler bu blok içine yazılır.

Set Bloğu: Değişken içine değer yazdırdığımız zaman çalıştırılan bloktur.

Property: Get ve Set bloklarının içinde yer aldığı ve değişkenlerin kontrolünün sağlandığı yapıdır.

Nesne içindeki verileri gizleme işlemine kapsülleme denilmektedir.

Kapsülleme kavramını daha iyi anlayabilmek için araç sınıfımızı biraz daha genişleterek örneğimize devam edelim. Aracımızda bazı alanların değiştirilebileceğini gözlemledik. Şimdi ise bazı alanların sadece okunabilir bazı alanların da ne okunabilir ne de yazılabilir olduğu bir örnek üzerinde anlatalım.

Araç sınıfının özellikleri;

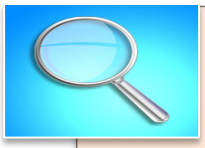
- Motor (erişilebilir)
- Direksiyon (erişilebilir)
- Teker (erişilebilir)
- Şasi (gizli)
- Camlar (sadece okunabilir)

İnsanların kimlik numarası gibi her araç için özel olan şasi numarasını gizli olarak belirledik. Bununla beraber camlar üzerinde oynama işleminin gerçekleştirilmemesi için (özellikle ön cama film takılmaması gibi bir örnek olabilir) sadece okunabilir olarak ayarladık. Şimdi araç sınıfının özelliklerini kalıtımla alan araba sınıfından Mercedes nesnesi üzerinde örneğimize devam edelim.

Araba sınıfına ait Mercedes nesnesinin özellikleri;

- Motor = 3200; (kalıtımla gelen değiştirilebilir özellik)
- Direksiyon = Fonksiyonel; (kalıtımla gelen değiştirilebilir özellik)
- Teker = 4 tekerli; (kalıtımla gelen değiştirilebilir özellik)
- Camlar = Arka camlar karartılmış (Kalıtımla gelen gizli ama okunabilir özellik)
- Model = “S320”; (Araba sınıfına ait değiştirilebilir özellik)
- Kasa = “Sedan”; (Araba sınıfına ait değiştirilebilir özellik)

Örneğimizi incelediğimiz zaman Şasi hariç diğer özelliklerimizi görebiliyoruz. Şasi özelliği nesne üretilirken yazılan ve okunup değiştirilmesi istenen bir özellik olmadığı için gizlenmiştir. Böylelikle kapsülleme sayesinde istenilen özellikler değiştirilebilir ve bazı özelliklerde kalıtımla gelmiş olsa dahi değiştirilemez olarak ayarlanabilmektedir.



Özet

- Nesneleri kullanarak gerçekleştirilen programlama anlayışı yaklaşık olarak 50 yıldır var olsa da, nesneler son 20 yılda programlamanın vazgeçilmez bir öğesi haline gelmiştir. Günümüzde sıklıkla kullanılan nesne yönelimli programlama anlayışından bahsedebilmek için programlama dilinde aramamız gereken bir takım kavramlar bulunmaktadır. Bu kavramlar; Sınıf, Nesne, Kalıtım, Çok biçimlilik ve Kapsüllemedir.
- Sınıf kavramını günlük hayatımızla ilişkilendirecek olursak, gerçek dünyada gözlemlediğiniz bazı nesnelerin soyut olarak ifade edilmesi veya basitleştirilmesi olarak tanımlayabiliriz. Nesneler ise bu sınıfların birer üyesidir. Bir sınıfı temel olarak iki bileşene ayırmamız mümkündür. Bunlar; nesneyi tanımlayan özellikler ve nesneyle ilişkilendirmek istediğimiz yöntemler veya eylemlerdir.
- Bir nesneyle ilişkilendirmek ve kaydetmek istediğiniz veriler, sınıfın özelliklerini oluşturmaktadır. Sınıf özelliklerinin değerlerini değiştirerek nesnenin durumunu değiştirebilirsiniz. Bir nesnenin durumu, nesneyi tanımlamak için kullanılan özelliklerin değerleriyle belirlenir.
- Bir nesneye ait durumu tanımlamak için özellik değerleri kullanılmaktadır. Sınıf metotları ise bu özelliklere etki etmektedir. Buradan hareketle metot kavramını, nesnelerin belirli işlemlerini gerçekleştirmeyi sağlayan fonksiyonlar olarak tanımlayabiliriz. Kısaca sınıf metotları, nesnenin gerçekleştirebileceği davranışları veya eylemleri belirlemektedir.
- Metotlar genel olarak değer (parametre) alabilen, değer döndüren ve birden fazla değer ile işlem yapabilen yapıdadırlar. Metotları nesnelerle ilişkilendirmek istediğimiz eylemleri tanımlamak için kullanırız.
- Hazırladığımız programlarda gerçekten kullanabileceğimiz yapılar nesnelerdir.
- Öncelikle nesnenin türetileceği sınıf adı yazılır ve bir boşluk bırakıldıktan sonra türetilecek olan nesnenin adı yazılmaktadır. Ardından eşittir sembolü kullanılarak New anahtar kelimesiyle birlikte sınıfın adı tekrardan yazılır. Böylelikle belirtilen sınıfa ait bir nesne türetilmiş olur. Burada kullanılan New anahtar kelimesi ile sınıftan nesne türetilmiştir. Burada kullanılan metoda Yapıcı Metot ve Constructor denilmektedir. Bu metotların detaylarına ilerleyen ünitelerde değinileceği için şimdilik bu kadar bilgi sahibi olmanız yeterli görülmektedir.
- Nesne türetme işlemi gerçekleştirilirken nesne adını belirlediğimiz alanda belirli bir yazım tarzı benimsememiz hazırladığınız kodların okunabilirliğini olumlu yönde etkileyecektir. Bununla beraber isimlendirme kurallarını da dikkate almanız gerekmektedir.
- Kalıtımdan bahsedebilmek için bir sınıf kendinden üstte bulunan sınıfın özelliklerini taşıması gerekmektedir. Böylelikle bir sınıfa ait özelliklerin bir veya daha fazlasını başka sınıf veya sınıflar içinde kullanabiliriz. Projelerimizde kalıtımı kullanabilmek için iki nokta üst üste (:) kullanmamız gerekmektedir. Bu işlem ":" operatöründen önce sınıfımızın ismini, sonra da kalıtım alınacak sınıfın ismini yazarak gerçekleştirilmektedir.
- Kalıtımla bir sınıftan alınan özellik olduğu gibi kullanılıyorken, çok biçimlilik işleminde bu özellik farklı şekillerde de kullanılabilmektedir. Bu işleme aynı zamanda Polimorfizm de denilmektedir. Çok biçimlilik durumunu örneklemeden önce birkaç teknik terimden kısaca bahsedilecektir.
- Virtual: Bir sınıftan alınan kalıtımının değiştirilebilmesi için kullanılan komut.
- Override: Bu komut ise kalıtım ile gelen metodun bulunduğu sınıftaki değer yerine başka bir değer kullanılarak işletilmesini sağlamaktadır.
- Komutlar ve tanımlarından da anlaşılacağı üzere çok biçimlilik, kalıtımla gelen metotların farklı biçimlerde kullanılmasına imkan tanıyan bir özelliktir.
- Kapsülleme nesne yönelimli programlamanın önemli özelliklerinden birisidir. Bazı durumlarda nesnelerin özelliklerini gizlemek zorunda olabiliriz. Yani daha öncede bahsedildiği gibi nesne adı yazıldıktan sonra nokta operatörü konularak açılan özellikler listesini kısıtlamak isteyebilirsiniz. Biraz daha açacak olursak, verileri programın diğer bölümleri tarafından yanlışlıkla yapılan değişikliklerden korumak için bir nesnenin içine yerleştirirsiniz. Bu şekilde yapılan bir nesne içindeki verileri gizleme işlemine kapsülleme denilmektedir.

DEĞERLENDİRME SORULARI

Bir nesnenin durumu, nesneyi tanımlamak için kullanılan ……. değerleriyle belirlenir.

1. Cümlede boş bırakılan yere aşağıdakilerden hangisi getirilmelidir?
   a) Özelliklerin
   b) Verilerin
   c) Çıktıların
   d) Sınıfların
   e) Kayıtların

2. Aşağıdakilerden hangisi genel olarak değer (parametre) alabilen, değer döndüren ve birden fazla değer ile işlem yapabilen yapılardır?
   a) Kontroller
   b) Metotlar
   c) İsim uzayları
   d) Private
   e) Public

3. Aşağıdakilerden hangisi günlük hayatta yaptığımız planlama işinin programlama anlamında karşılığına denk gelen terimdir?
   a) Kapsülleme
   b) Çok biçimlilik
   c) Algoritma
   d) Kalıtım
   e) Erişim

4. Aşağıdakilerden hangisi bir sınıfa ait özellikleri kullanmamızı sağlayan yapıdır?
   a) Sınıf
   b) Algoritma
   c) Kod
   d) Nesne
   e) Dosya

5. Aşağıdaki kod satırlarından hangisi Araba isimli bir sınıftan Mercedes isimli bir nesne türetme işlemine ait kod satırını doğru olarak vermektedir?
   a) Araba = Mercedes
   b) Mercedes = Araba
   c) Araba New Mercedes
   d) Araba new = Mercedes
   e) Araba Mercedes = New Araba

6. Aşağıda yapılan tanımlamalardan hangisi isimlendirme kurallarına göre doğru değildir?
   a) Araba1
   b) Araba_
   c) ArabaAnd
   d) araba
   e) araba and

7. Projemize yeni bir sınıf eklemek için aşağıdaki menü yollarından hangisini kullanmamız gerekir?
   a) Derle – Sınıf Ekle
   b) Görünüm – Kod
   c) Proje – Sınıf Ekle
   d) Derle – Çözümü Derle
   e) Git – Kopyala

8. Bir nesnenin özelliğini kullanabilmek için yazmamız gereken söz dizimi aşağıdakilerden hangisinde doğru olarak verilmiştir?
   a) Nesne_adı.özellik
   b) Nesne_adı = özellik
   c) Özellik.Nesne_adı
   d) Özellik = Nesne_adı
   e) Nesne.özellik.Nesne_adı

9. Bir sınıftan kalıtım yoluyla başka bir sınıf türetebilmek için aşağıdaki operatörlerden hangisini kullanmamız gerekir?
   a) Noktalı virgül(;)
   b) İki nokta üst üste (:)
   c) Virgül (,)
   d) Nokta (.)
   e) Çift tırnak (“”)

10. Bir sınıftan kalıtımla alınan bir özelliğin farklı şekilde kullanılabilme durumu aşağıdakilerden hangisiyle ifade edilir?
    a) Çok biçimlilik
    b) Kalıtım
    c) Kapsülleme
    d) Sınıf
    e) Örnekleme

Cevap Anahtarı
1.a, 2.b, 3.c, 4.d, 5.e, 6.e, 7.c, 8.a, 9.b, 10.a

YARARLANILAN KAYNAKLAR

Aktaş, V. (2010). *Visual studio 2010 ile her yönüyle C# 4.0* (2. Baskı). İstanbul: Kodlab Yayın.

Aktaş, V. (2020). *Visual studio 2019* (1.Baskı). İstanbul: 01 Yayınları.

Atasever, V. (2017). *C# 7 Uygulamalarla C# programlama dilini keşfedin* (2. Baskı). Kocaeli: Level Kitap.

Demirli, N. & İnan, Y. (2005). *Visual C# .Net* (3. Baskı). Ankara: Palme Yayıncılık.

Purdum, J. (2013). *Beginning Object-Oriented Programming with C#.* Indianapolis: John Wiley & Sons, Inc.

Sharp, J. (2011). *Adım adım Microsoft visual C# 2010*. (Çev. T. Buldu). Ankara: Arkadaş Yayınevi.

Uzunköprü, S. (2017). *Projeler İle C# 7.0 ve SQL Server 2016* (7. Baskı). İstanbul: Kodlab Yayın.

Vatansever, F. (2004). *Algoritma Geliştirme ve Programlamaya Giriş* (2. Baskı). Ankara: Seçkin Yayıncılık.

Yanık, M. (2008). *Visual studio 2008 ile Microsoft visual C# 3.0 for .Net framework 3.5* (1. Baskı). Ankara: Seçkin Yayıncılık.

Yücedağ, M. (2020). *C# Eğitim kitabı* (3. Baskı). İstanbul: Dikeyeksen.