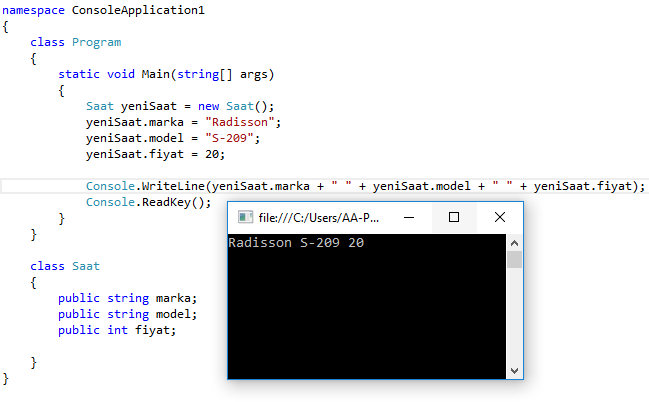
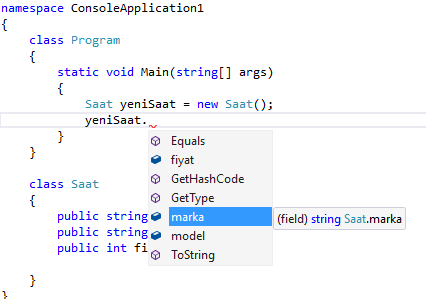
# Encapsulation (Sarmalama)

Konuya giriş yapmadan önce field ve property kavramları hatırlanmalıdır. Bir göz atmak gerekirse;

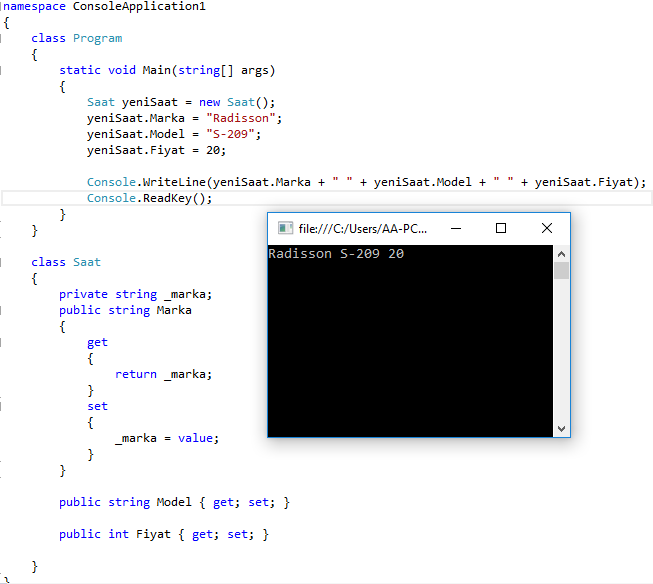
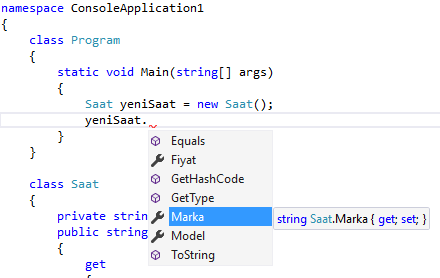
**Field**: Class içindeki *değişkenlerdir* diyebiliriz. Değişken özelliklerinin tamamını sağlarlarlar.

**Property**: *Field’ları yöneten* özelliklerdir. Get ve Set methodları vardır. Belirlenen kurallara ve özelliklere göre yönettikleri field’lara değer ataması yada field’daki değeri çekme işlemi yaparlar.

Field ile Class:



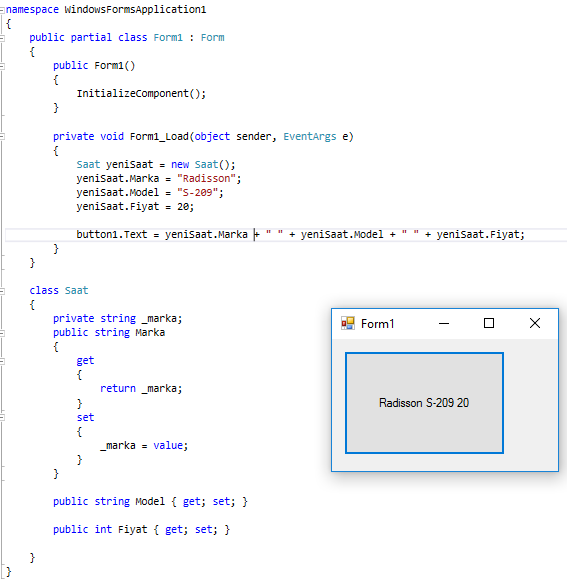
Property ile Class :



*(EĞİTMEN NOTU: Burası zaten 28.Gün anlatılıyor. O nedenle öğrenciye sorular sorarak hızlı ve interaktif bir şekilde geçilebilir. Burada debug üzerinden field ve property nin nasıl işlediği gösterilmeli! )*

*(EĞİTMEN NOTU: Windows From konusuna giriş yapıldığından bu konunun Windows Forms üzerinden anlatılması gerekir. Görseller eski, yeni versiyonu :*

*)*



Artık encapsulation konusuna giriş yapılır.

Field ve property arasındaki ilişki şu şekilde düşünülebilir :

Bir telefonun iç aksanı kapalıdır. Kullanıcıdan gizlenir. Kullanıcının bilemesi gereken tek şey ön tarafındaki ekranda gördüğü arayüzdür. Orada dilerse rakamlar girip arama yapabilir. Arama yapılırken aranacak numaraların hangi değişkenlerde tutulduğu, bunun ram de hangi transistörler üzerinde bilgilerinin yer aldığını, kaç mHz frekansla haberleşme sağlayıp aramayı gerçekleştirdiği ve anteninin nerede olduğu gibi bilgileri kullanıcının bilmesine gerek yoktur. Amaç ise yanlış kullanımı önlemek ve ürün sağlığını korumaktır. Bir classda fieldlar iç aksandır. Propertyler ise iç aksan ile arayüz arasındaki ve geçişin yönetiminden sorumludur.

Yukarıdaki class üzerinden incelersek, nesne kullanılırken Marka propertysine değer gönderilir. Gönderilen değer ise property üzerinden field a aktarılır. İşte bu işlem **Encapsulation** işlemidir aslında. OOP için temek karakteristiklerdendir.

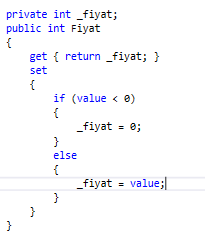
Tanımlamak gerekirse **Encapsulation (Sarmalamak-Kapsüllemek) c**lass içindeki Field lara dış dünyadan kontrollü bir şekilde geçiş yapmaktır ve bu kontrol property ler üzerinden yapılır.

*(EĞİTMEN NOTU: Anlatılanları pekiştirmek için şu soru üzerinden derse devam edilebilir:*

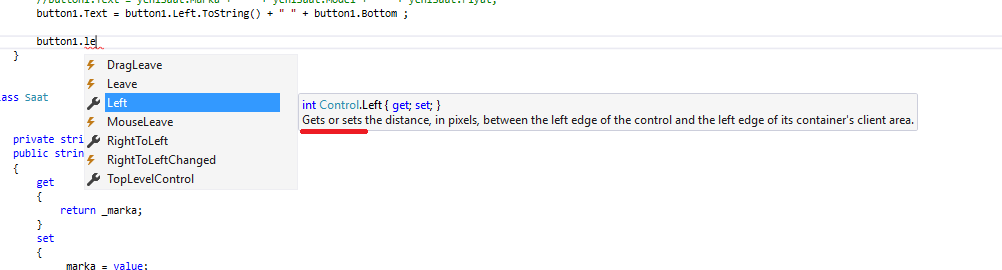
*Yukarıda property içeren classlar ile saat bilgisini buton textine atabildim. Sadece fieldlar ile class oluşturunca da aynısını yapabiliyorum. Peki neden property yaptık o zaman?*

*Burada cevap olarak Fiyat özelliğine mantıksız bir girdi olan negatif bir sayı girilir. Bunun önlenmek istendiği ve nerde yapılabileceği sorulur. Düşünmelerine yardımcı olmak için ise butonun genişliğine negatif bir değer verilerek debug ile içinde ne olduğuna bakılır.*

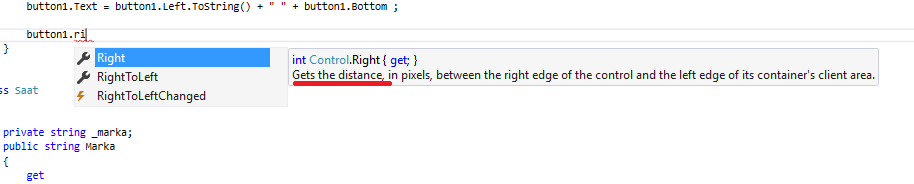
*Bundan sonra ise OOP özelliklerinden “reusability (tekrar kullanılabilirlik)” özelliğine dem vurularak bunun class içinde olması gerektiği anlatılır ve class içinde set blogunda negatif girilirse fielda 0 değerini aktaran kodlar eklenir.*

**

*Fieldların private olmasının sebebi eğer field dışardan erişilebilir olsaydı bu kontrolden geçmeden direk olarak field a negatif bir değer girilebilir olur. Bunu önlemek istenir. )*

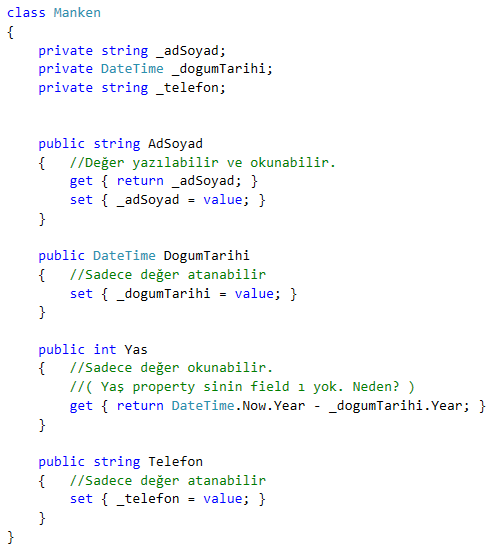
Butonun left özelliğine baktığımızda get ve set edilebilir yani değer atanabilir ve değer okunabilir olduğunu görürüz:

Ancak bazı özelliklere değer atanması istenmeyebilir:



Yada bunun tersi olan değer alınması durumu da engellenmek istenebilir. Bu durumu ufak bir örnek class ile özetlemek gerekirse;

Elimizde ünlü mankenlerin bilgilerini modelleyeceğimiz bir class olsun. Bu classın propertyleri aşağıdaki gibidir:



Yukarıdaki class içerisini incelemek gerekirse; AdSoyad property si get ve set edilebilir yapıdadır. Yani kontrol ettiği field a hem değer atılabiliyor hem de değer okunabiliyor. Zaten mantık olarak AdSoyad bilgisinde gizlemek istenilen bişey yoktur.

Ancak doğum tarihi konusunda aynı durum geçerli değil. Manketlerin doğum tarihi bilgisine doğrudan ulaşılması istenmez. Sisteme girişi için sadece set blogu açıktır. Herhangi bir yerden erişim olmaması için get bloğu kaldırılmıştır. Bu bilgi için sadece get edilebilen ve dogum tarihine bağlı olarak hesaplanan Yas propertysi kullanılmıştır. Sistem saatinin yılından doğum tarihinin yılını çıkartarak bulunur.

Telefon bilgisi için de benzer bir durum söz konusudur. Telefon bilgisine doğrudan ulaşılmaz. Sadece sisteme girişi yapılır.

Burada DogumTarihi ve Telefon propertyleri **write only (sadece yazılabilir)** propertydir, Yas propertysi ise **read only (sadece yazılabilir)** propertylerdir.

*(EĞİTMEN NOTU: Son olarak burada field property ilişkisini pekiştirmek amaçlı “Yaş propertysinin neden field ı yok?” diye sorulabilir)*

# Inheritance

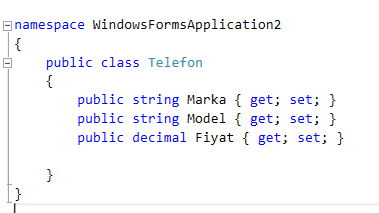
OOP için temel karakteristiklerden biri olan **Inheritance** Türkçeye kalıtım olarak çevrilebilir. Kalıtım kelimesi daha çok babadan oğula geçen özellikler olarak bilinir. C# için OOP tanımında da bir class başka bir class a kalıtım verebilir yani özelliklerini aktarabilir. Kalıtım veren class “base class”, kalıtım alan class ise “derived class” olarak adlandırılır. (base : temel , derived : türetilmiş).

**NOT**: Bir class birden fazla class a kalıtım verebilir. Ancak C# için bir class ın ancak bir tane kalıtım veren yani base class ı olabilir. Yani bir babanın birden fazla çocuğu olabilir ancak bir çocuğun tek babası olur.

## Nasıl Kullanılır?

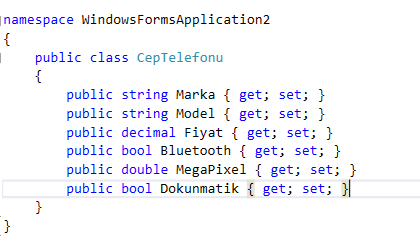
Aşağıdaki örnek classlar üzerinden incelenmelidir.

*Öncelikle “Telefon” adında bir class oluşturalım.*

**

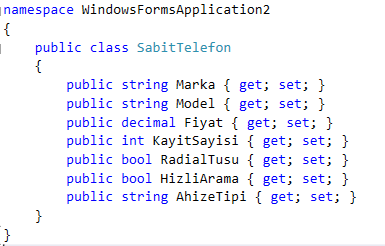
*Yukarıdaki propertyler otomatik encapsule edilmiş olarak yazılmıştır. Fieldları gizlidir. Bundan sonraki örneklerde gerekmedikçe get set methodlarını açmayacağız.*

Şimdi yeni bir class oluşturalım. Bu class da “CepTelefonu” classı olsun.



Cep telefonunda olması istenilen özellikler ile oluşturulan class da yukarıdaki gibidir.

Bir de “SabitTelefon” class’ı oluşturulur:

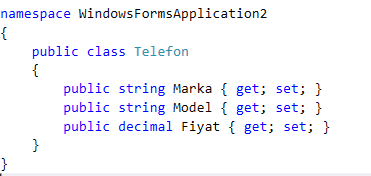


Sabit telefonda olması istenilen özelliklerle oluşturulan class da yukarıdaki gibidir.

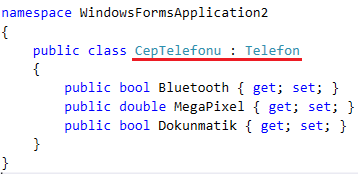
Burada her iki classda da olan “Marka”, “Model” ve “Fiyat” özellikleri telefon türü farketmeksizin ortak olan özelliklerdir. Bu propertyleri tekrar tekrar her telefon çeşidinin classlarına yazmak mantıksız olur. **Inheritance** burda devreye girer. Bu özellikler “Telefon” adında başka bir classa alınır. Sonrasında da “SabitTelefon” ve “CepTelefonu” classlarına kalıtım verilir. Boylece “Telefon” classındaki özellikler diğer classlara da geçer.

Inheritance karakteristiği ile oluşturulan yeni yapı şu şekilde olur.

Telefon class’ı:

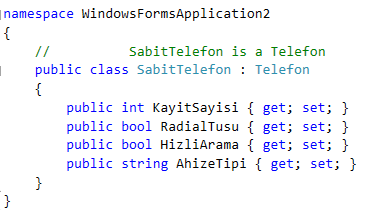


Telefon Class’ından kalıtım alan CepTelefonu class’ı ise;



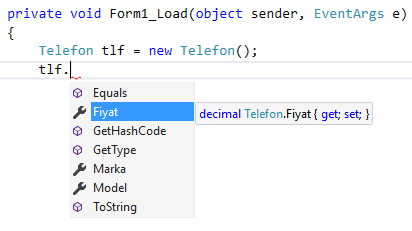
C#da bir classa kalıtım vermek için, kalıtım alınacak class adının yanına iki nokta koyulur ve kalıtım verecek class yazılır. Yukarıdaki örnekde base class “Telefon” iken derived class “CepTelefonu”dur.

“SabitTelefon” class :

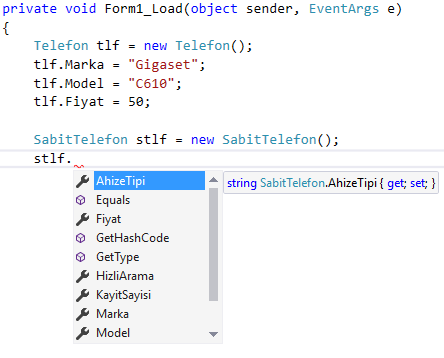


Inheritance karakteristiği ile yeniden düzenlenen SabitTelefon class’ın son hali bu şekildedir. Burada aslında SabitTelefon bir Telefondur. Telefon classından türetilmiştir.

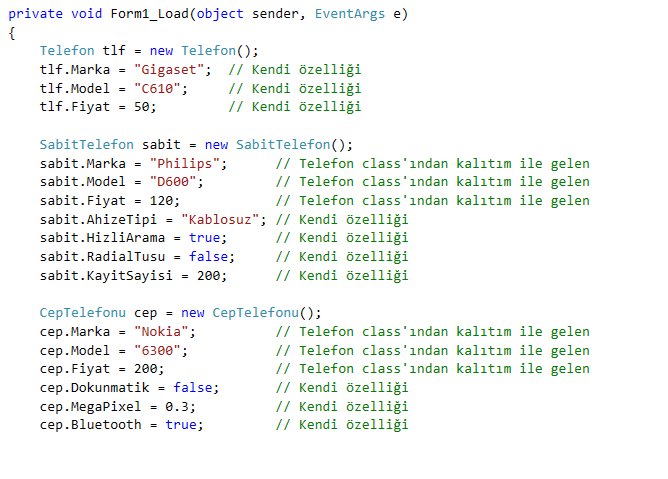
Windows Form içerisinde kullanımına bakılırsa;



Derived Classlar olan CepTelefonu ve SabitTelefon için de kullanıma bakılırsa:



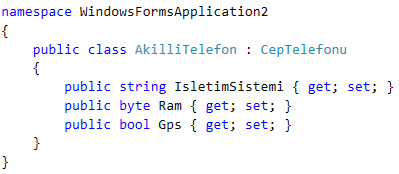
Görüldüğü gibi “SabitTelefon” classına ‘Marka, ‘Model’ ve ‘Fiyat’ özellikleri yazılmamasına rağmen telefon classından kalıtım yoluyla bu özellikler elde edildi. Yukarıda yazdığımız classların nesnelerinin son hali ise şu şekildedir;



Örnekteki seneryoya devam edilecek olursa yeni bir “AkilliTelefon” class’ı üretmek istenmektedir. Bu class da “Telefon” classından türetilebilir. Ortak özellikler “AkilliTelefon” classına da aktarılabilir. Ancak “AkilliTelefon” classının cep telefonu classındaki özelliklere de ihtiyacı vardır.

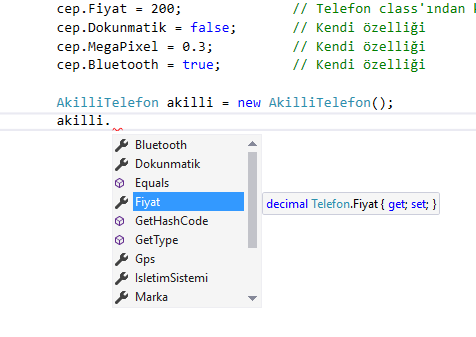
Bu noktada “AkıllıTelefon” classını “Telefon” classından değil de “CepTelefonu” classından üretmek mantıklıdır. Derived classlar başka classların base classları olabilirler. Kendi derived classlarına aktarabilirler. Bu şu demek; ben babamdan aldığım özellikleri oğluma aktarabilirim. Sebebi ise babamdan almış olsam bile o özellikler aynı zamanda benim de.

Şimdi ise C# ın OOP penceresinden class yapısına ve kullanımına bir bakmak gerekirse;



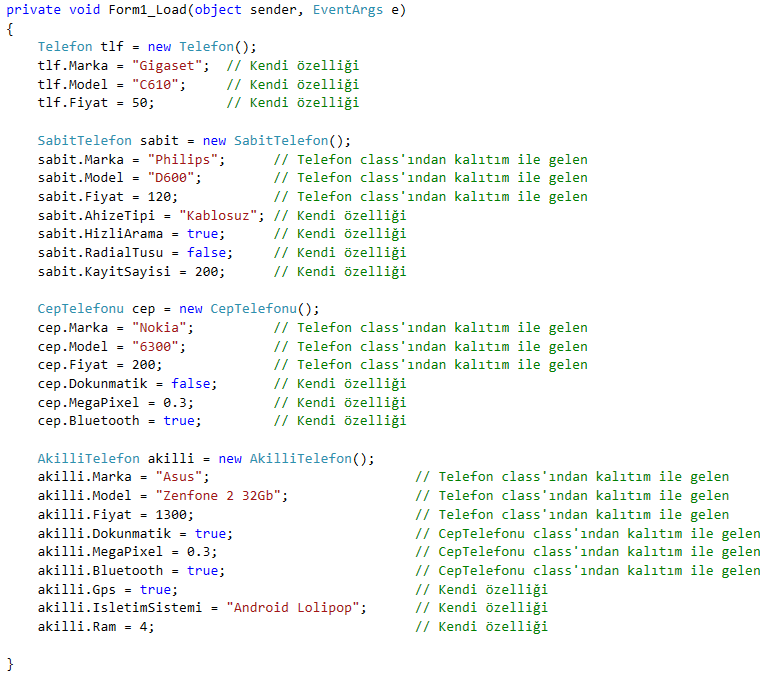
Yeni olusan “AkilliTelefon” classının nasıl kullanıldığına ve kaç özelliğin geldiğine bakmak gerekirse;

*(EĞİTMEN NOTU: Burada öğrenciye hangi özelliklerin geleceği sorulabilir. Sadece CepTelefonu classının içine yazdığımız özellikler mi yoksa Telefon classından CepTelefonuna aktarılan özelliklerin de gelip gelmeyeceği sorulabilir.)*



Yukarıda görüldüğü gibi “AkilliTelefon” classı “CepTelefonu” classından kalıtım almış olsa bile ‘Fiyat’ propertysi “CepTelefonu”na “Telefon” classından gelir ve “CepTelefonu” classı da bu özellikleri “AkilliTelefon” classına aktarır.

Son durumda ise bütün üretilen classlardan oluşturulan nesnelerin kullanımları ise şu şekildedir:



**~~Encapsulation~~**

~~Encapsulation (kapsülleme) nedir?~~

~~get set neden yazılır?~~

~~read-only write-only~~

**Inheritance**

~~Inheritance nedir?~~

~~nasıl kullanılır?~~

protected üye kullanımı???