

BİL 288 BİLGİSAYAR PROGRAMLAMA-II

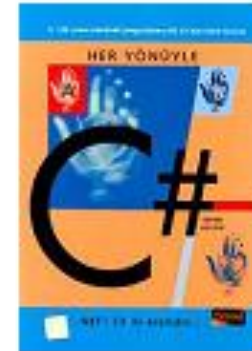
C

Neler Öğreneceğiz

- Nesneye yönelik programla nedir?
- (Metot nedir?, sınıf nedir?, nesne nedir?, nesneler nasıl türetilir?.....)
- Nesneye yönelik programlamanın üstünlükleri nelerdir ?
- .NET teknolojisi nedir? Neleri bizlere sağlar ve avantajları nelerdir?

Kaynaklar

- C#'ı Kavramak
 - Tom Archer, Arkadaş, 2002
- Her yönüyle C#
 - Sefer Algan, Pusula, 2004
- C# Temel Başlangıç Kılavuzu
 - BradleyJones, Sistem, 2003
- www.bilgeadam.com



C# Nedir?
.Net Framework Nedir?

C# Nedir?

Bilgisayar dillerini düzeylerine göre şu şekilde sınıflandırabiliriz:

Script Dilleri	Javascript, VBScript, Perl Script
Yüksek Düzeyli Diller	Vbasic, Delphi
Orta Düzeyli Diller	C# ve Java
Düşük Düzeyli Diller	C/C++
Assebmly(makine dili)	Assembly

C# Nedir?

- C# Programlama Dili Microsoft'un son zamanlarda geliştirdiği .NET platformunun bir ögesidir.
- Eski programlama dilleri ile yeni dillerin harmanlanması ile oluşmuştur.
- Ayrıca C, C++, Java, Visual Basic dillerinin bir türevi niteliğindedir.

C# Nedir?

- **C#** , C/C++ ve Java dillerinden türetilmiş, bu dillerin dezavantajlarının elenip iyi yönlerinin alındığı, güçlü basit, esnek, tip güvenli (type safe) Net platformu için sıfırdan geliştirilmiş %100 nesne yönelimli bir dildir.
- (type-safe, tür dönüşümlerindeki önlemler, örn: `byte=byte+byte` olamaz, `int=byte+byte`).

C# Nedir?

- C#, eskiden beri programcıların yaygın bir biçimde kullandığı C/C++ve Java dillerine benzerliği ile tanınan bir programlama dilidir. İlk aşamada çok benzer bir dil olarak görünse de bu iki dilden farklı bir çok özelliğe sahiptir.
- C/C++ve Java'nın güzel özelliklerini alıp bu dillerin tehlikeli olabilecek olabilecek özelliklerini dışarıda bırakan bir dildir.

C# nedir?

- C#, C/C++ dilinden farklı olarak tamamıyla nesneye yöneliktir. “int”, “double” gibi temel veri türleri dahi birer nesne olarak tanımlanmıştır.
- Java dilinden farklı olarak C# dilinde işaretçiler (**pointer**) kullanılabilmektedir.

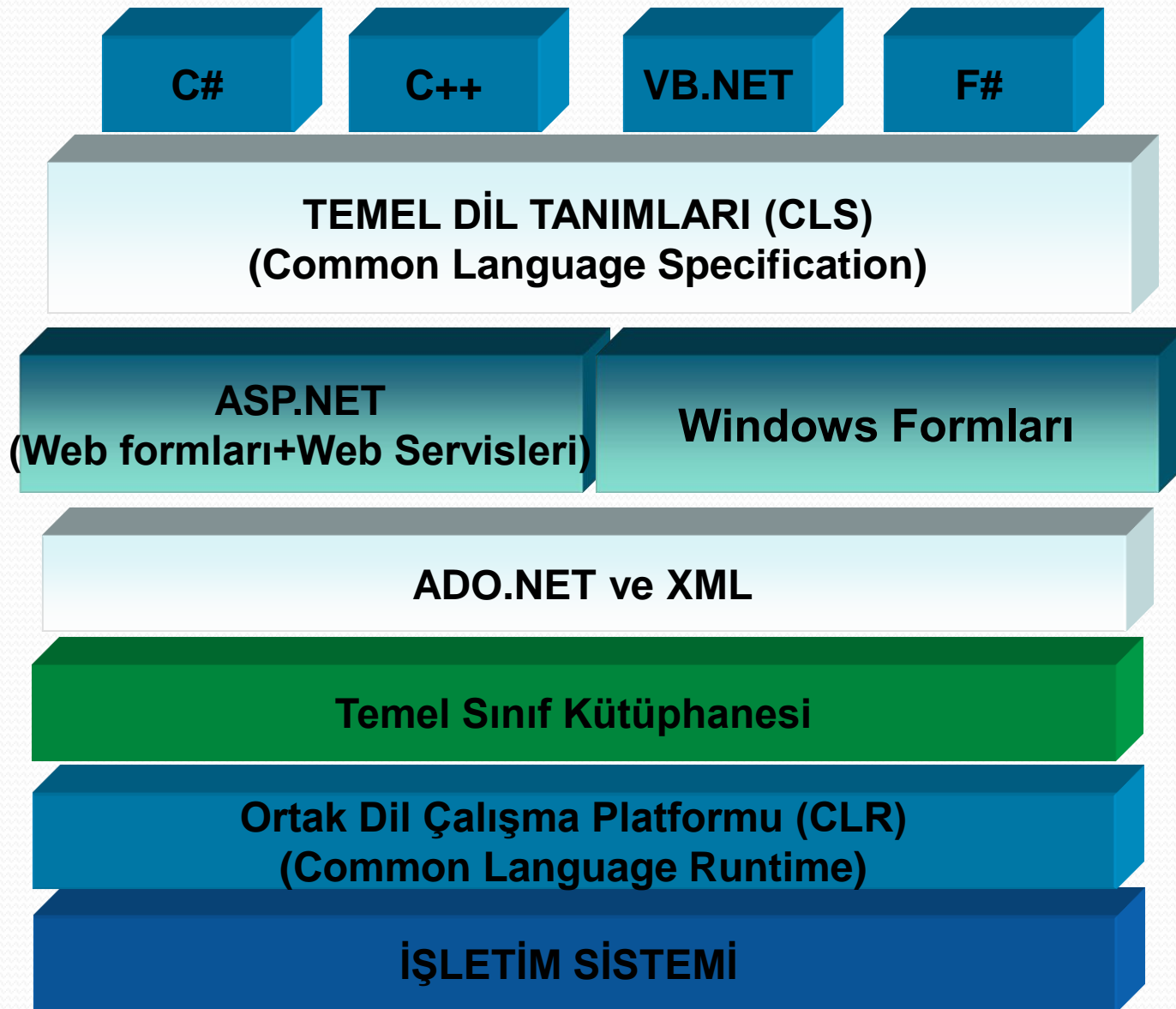
Neden C#'ı Tercih Edelim?

- Öğrenilmesi kolaydır.
- **Nesne yönelimli programlamaya tam destek vermektedir.**
- Yüksek verim.
- Güç ve kolaylık arasında dengededir.
- Xml desteği sunmaktadır.
- Windows Forms mantığını destekler.
- İnternet teknolojilerinin gelişimine uyum sağlayabilmiş modern bir dildir.

C# Kullanım Alanları

- **Konsol tabanlı uygulama geliştirme**
- **Windows için program yazma**
 - *C# ile Windows için gelişmiş, güçlü, hızlı ve güvenli programlar yazılabilir. Fakat bunun için programın çalıştığı sistemde .NET platformunun yüklü olması gerekir.*
- **ASP.NET için en uygun dildir.**
 - *ASP.NET, ASP gibi script yorumlamalı olmayıp tamamen nesne yönelimli haldedir. C# bu konuda büyük kolaylıklar sağlamaktadır.*
- **C# ile Web Servisleri geliştirmek oldukça kolay ve hızlıdır**
- **Mobil uygulama geliştirme ve DLL yazabilme.**

.NET Bileşenleri



Visual Studio NET 2010

.NET Bileşenleri

- Önceden, geliştirilen yazılımlar direkt olarak makine koduna derlenir ve bu şekilde çalıştırılırlardı. Ayrıca bu programlar, her işletim sistemine özel olarak geliştirilmekte ve derleme işlemi işletim sistemine göre belirlenmekteydi.
- Böyle bir yapıda taşınabilirlikten söz etmek mümkün değildir.
- **Java'da ise, program önce byte code'a çevrilmiştir. Bu kodu JVM(Java Virtual Machine), işletim sisteminin istediği koda çevirmektedir.**
- .NET platformunda da temel prensip Java ile benzerdir.

.NET Bileşenleri

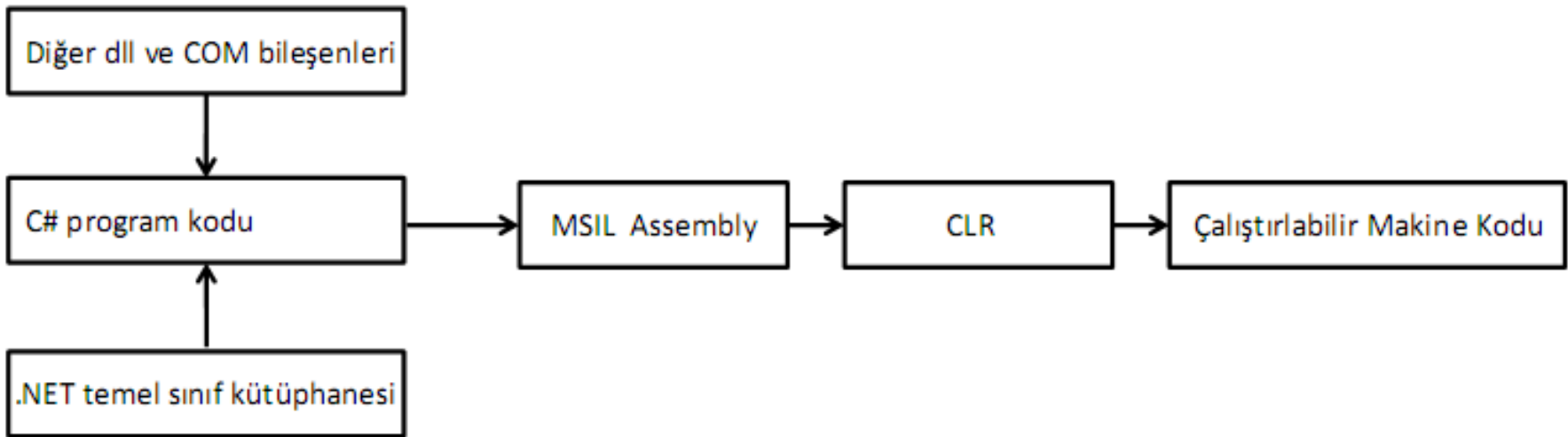
- . NET kodu önce **IL** (Intermediate Language-Ara dil)'ye derler ve bu IL kodu çalıştırılmak istendiği zaman **.NET CLR** (Common Language Runtime-Ortak Çalışma Platformu), JIT (Just In Time) derleyicilerini kullanarak makine diline çevirir.
- CLR makine diline çevrilmiş bu kodu önbellekte tutar, bu performans artışına sebep olurken diğer taraftan sistem hafızasında küçümsenmeyecek yer işgal eder.

.NET Bileşenleri

- Temel Dil Tanımlamaları **CLS** (**C**ommon **L**anguage **S**pecifications) ve Ortak Tip Sistemi **CTS** (**C**ommon **T**ype **S**ystem) ile .NET uyumlu dillerin hepsi aynı değişkenleri ve benzer nesne yönelimli özellikleri taşır. Örn: C# ile yazılan programdaki temel veri tipleri, VB.NET 'tekiler ile aynı özelliklere sahiptir. Böylece farklı dillerde yazılan bileşenler birbiri ile sorunsuz çalışırlar.

Ortak Dil Çalışma Platformu(CLR)

- **CLR**, .NET altyapısında programların çalışmasını kontrol eden ve işletim sistemi ile programımız arasında yer alan arabirimdir. (Normalde yazılan kodlar makine diline çevrilir ve işletim sistemi ile direkt bağlantı kurup çalışırdı.)



- IL kodu, CLR tarafından çağrılınca JIT derleyicileri tarafından makine diline çevrilir ve çalıştırılır.

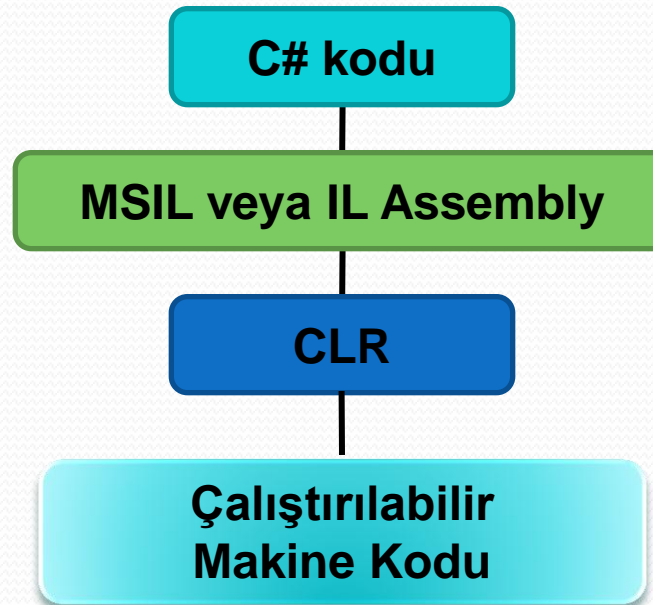
Ortak Dil Çalışma Platformu(CLR)

- Eğer çok sayıda platform olduğunu düşünürsek, programların bunlar için ayrı ayrı yazılıp derlenmesi gerekir. Bu durum imkansız gibidir.
- Eğer platformdan bağımsız bir ortam istiyorsak, ihtiyaç duyulan şey CLR dir, hangi platformda iseniz (Linux,Mac,Windows) CLR bu noktada devreye girer ve .NET programlarının farklı platformlarda işletim sistemine göre çalıştırır.

Ortak Dil Çalışma Platformu(CLR)

- **Managed Code(Yönetilen Kod):** Yalnızca CLR yardımları altında çalışan koddur.
- Bir örnek vermek gerekirse ; Windows'ta çalışan farklı işlemlere sahibiz. Uygulamaların izlemesi gereken kural Windows genel kurallarına uymalarıdır. Managed kodda CLR tarafından Windows'un yaptığı şekilde çalıştırılan koddur.

Ortak Dil Çalışma Platformu(CLR)



.NET derleme ve çalıştırma

Aradil (IL veya MSIL) (Intermediate Language)

- Herhangi bir C++ veya Vbasic kodu direkt makine koduna çevrilirdi ve çalıştırılırdı. Makine diline çevrilen programlar, işlemciye ve işletim sistemine özel olarak derlenirdi.

Örn: a ve b sayılarının toplamı için kullanılan bir C++ programı Intel işlemciler için farklı, SunSparc işlemciler için farklı derlenirdi.

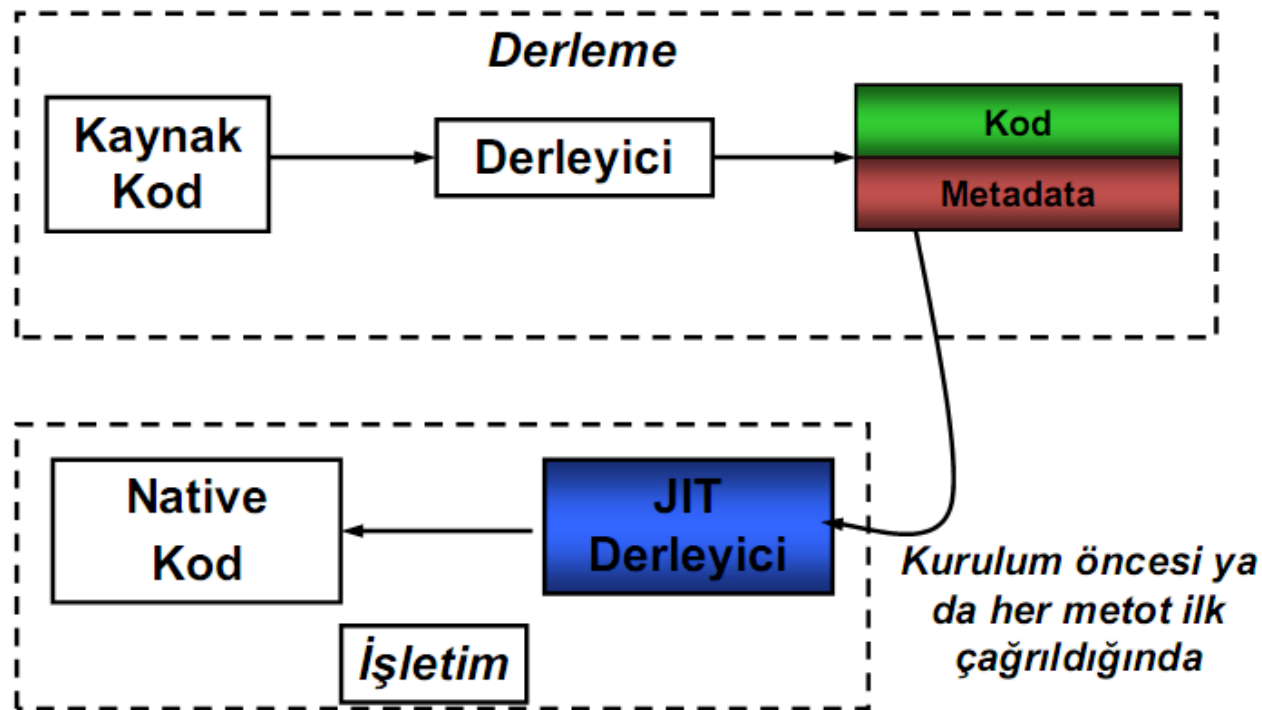
- Fakat .NET ortamında kodumuzu derlediğimizde elde ettiğimiz IL (ara dil) kodu işlemciye bağlı olmaz.

Aradil (IL veya MSIL)

- **IL** içerisinde değişken tanımları, değişkenlerin nasıl saklanacağı, metotların nasıl çalıştırılacağı, aritmetik ve mantıksal işlemler, bellek kullanımı gibi birçok işin nasıl yapılacağı açıklanır.
- Bütün bunların yanında IL'de **Metadata** olarak adlandırılan bir birim daha vardır. Metadata, programda kullanılan verilerin tiplerinin yanında oluşturulan sınıfların metotlarını ve bunların özelliklerini ve diğer bilgileri içerir.
- Artık IL ile oluşturduğumuz kodumuzun çalıştırılabilir bir program olması için derlememiz gerekiyor. Bunun için JIT (Just in Time) derleyici kullanılır.

JIT Derleyiciler (Just in Time)

- Metadata'nın içeriği çalışma zamanında JIT derleyicileri tarafından kullanılır IL ve Metadata'sı oluşturulan kod parçası, çalıştırılabilir bir yapıdır.



JIT Derleyiciler (Just in Time)

- C# ile IL'ye derlediğimiz programı çalıştırırken JIT derleyicileri devreye girerler. Bu derleyiciler programın çalıştırıldığı sistemin ve işlemcinin anlayabileceği makine kodunu oluştururlar.
- Windows ortamı için 3 çeşit JIT mevcuttur
 1. Normal JIT
 2. Pre-JIT
 3. Eco-JIT

JIT Derleyiciler (Just in Time)

- **Normal JIT** : IL kodu makine koduna çevrilirken default(varsayılan) olarak kullanılan derleyicidir. IL kodunu orijinal makine koduna çevirir ve *önbellekte* tutar. Örneğin ; program içindeki bir derlenmiş bir metot program akışı içinde tekrar çağrılırsa önbellekten çekilir.
- **Pre-JIT**: Tüm program kodunu makine koduna çevirip sonra çalıştıran JIT. Fazla hafıza gerektirir. Programın daha hızlı çalışmasını sağlar.

JIT Derleyiciler (Just in Time)

- **Eco JIT** : Kısıtlı hafıza ve önbellekli sistemlerde .NET programlarının daha iyi çalışmalarını sağlamak için kullanılan derleyicidir.
- Derlenen ve çalıştırılan program parçaları normal JIT'de hemen hafızadan silinmiyordu. Fakat Eco-JIT'de kullanılabilir hafıza belli bir oranın altına düştüğünde, daha önceden derlenmiş ve çalıştırılmış kısımlar hafızadan silinirler.

CTS

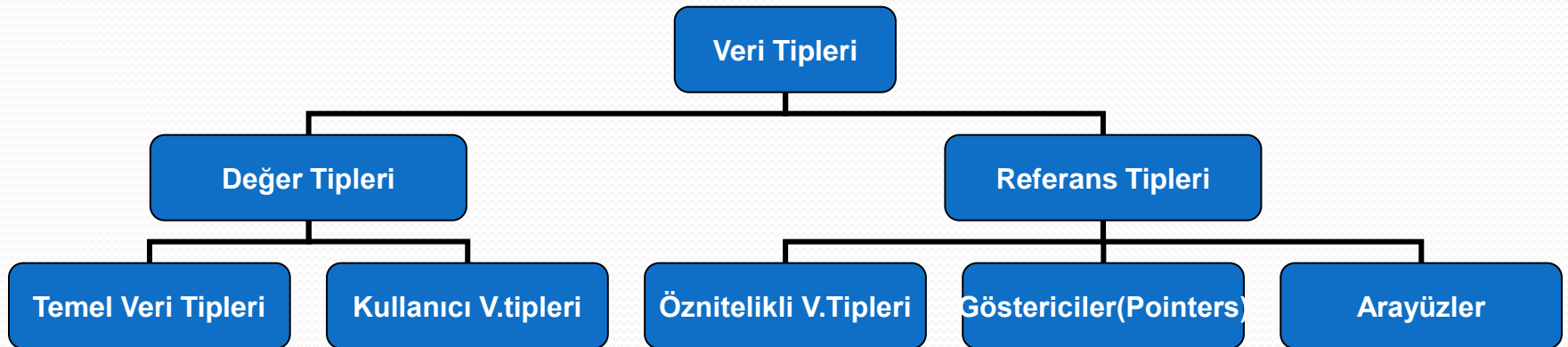
- Bütün veri tiplerinin tanımlı olduğu bir sistem olarak düşünebiliriz. C# dilindeki veri türleri aslında CTS'deki veri türlerine karşılık gelen ara yüzlerdir.
- CTS sayesinde .NET platformu için geliştirilen bütün diller aynı veri tiplerini kullanırlar, tek değişen türlerin tanımlama yöntemi ve söz dizimidir. Geliştirilen bir nesnenin diğer dillerde de sorunsuz çalışmasını garanti eder.

CTS

- **Örn:**
- Int tipi C++.NET ve VB.NET'teki tamsayı tiplerinin kapasiteleri aynıdır. Diğer bir deyişle VB'de geliştirilen bir dll C#'da rahatlıkla kullanılabilir.

CTS

- CTS sayesinde;
- .NET’de tip güvenli, yüksek performanslı ve kullanılan programlama dilinin diğer .NET uyumlu diller ile entegre bir şekilde çalışması sağlanır.
- Birçok programlama dilinin .NET mimarisinde tam nesne yönelimli olarak .NET için program yazma aracı olmasına imkan verir.
- Herhangi bir .NET uyumlu dilde geliştirilen nesne diğer dillerde de sorunsuz olarak çalıştırılabilir.



CTS veri tipleri şeması

CLS

- **CLS**, dil derleyicisinin uyması gereken kuralları içerir.
 - .NET platformunu paylaşan dillerin sadece CTS'yi desteklemeleri yetmemektedir. Bunun yanında **Ortak Dil Spesifikasyonu**' da (**CLS**) desteklemeleri gerekmektedir.
- **CLS**' ye uyan bir dille yazılmış kod ile diller arası iletişim sağlanmış olur.
 - CLS uyumlu bir dil ile geliştirilen bir program ile farklı diller arasında etkileşim sağlanabilir. .NET'in temel sınıf kütüphanelerinin içerisinde yer alan kodların büyük bir bölümü CLS uyumludur.

CLS

Desteklenen Programala Dilleri		
APL	Fortran	Pascal
C++	Haskell	Perl
C#	Java Language	Python
COBOL	Microsoft JScript®	RPG
Component Pascal	Mercury	Scheme
Curriculum	Mondrian	SmallTalk
Eiffel	Oberon	Standard ML
Forth	Oz	Microsoft Visual Basic®

Assembly

- .NET platformu için yazılan bütün kodların sonucunda oluşan .exe ve .dll uzantılı dosyalara genel olarak assembly denilmektedir.
- Derlenmiş kodlar ve metadata olarak adlandırılan özniteleyici kodlar Assembly içerisinde bulunurlar.
 - Assembly içerisindeki metadata verileri, tür bilgileri ve başka kaynaklara olan bağlantıları saklar.
 - Assembly'de ayrıca versiyon bilgisi de tutulur.
 - Assembly sayesinde programlar register edilmeye gerek kalmadan direkt kopyalanarak kurulabilirler.

Application Domain

- Application domain sayesinde aynı anda çalışan birden fazla program veya process birbirinden izole edildiği halde sistemde herhangi bir aksaklığa yol açmadan aralarında veri alış-verişi yapabilirler

Namespaces and .NET Class Library

(İsim Alanları ve .NET Sınıf Kütüphanesi)

- .NET Framework'ün programcılara sunduğu bir takım temel türler ve sınıflar mevcuttur.
- Bütün bu sınıfları ve türleri iyi organize edebilmek için .NET, isim alanı (**namespace**) kavramını kullanmaktadır.
- .NET'teki sınıf kütüphaneleri bir dilden bağımsız bir yapıdadır.

Namespaces and .NET Class Library

(İsim Alanları ve .NET Sınıf Kütüphanesi)

- C# dilinde .NET Framework sınıf kütüphanesi içerisindeki veri türleri ve sınıflar “**using**” anahtar sözcüğü ile kullanılır. Diğer dillerde de bu isim alanları farklı şekillerde derleyiciye bildirilir.
- Program geliştirirken sınıfların birbiri ile ilgili olanlarını aynı isim alanı içine konulması programdaki hataları bulma ve anlaşılabilirlik açısından oldukça önemlidir.

Namespaces and .NET Class Library

(İsim Alanları ve .NET Sınıf Kütüphanesi)

- .NET sınıf kütüphanesinde bulunan ve en sık kullanılan sınıf kütüphaneleri şunlardır:
- **System:** .NET ile çalışırken gerekli temel sınıfları içerir. Ayrıca diğer tüm sınıf kütüphaneleri bu isim alanı içinde kümelenmiştir.
 - Konsol temelli uygulamalarda temel giriş çıkış işlemleri için gerekli temel sınıf “Console”, bir çok matematiksel fonksiyonu içinde barındıran “Math” sınıfı da System isim alanı içerisinde yer alır.
 - System hiyerarşinin tepesinde bulunur.

Namespaces and .NET Class Library

(İsim Alanları ve .NET Sınıf Kütüphanesi)

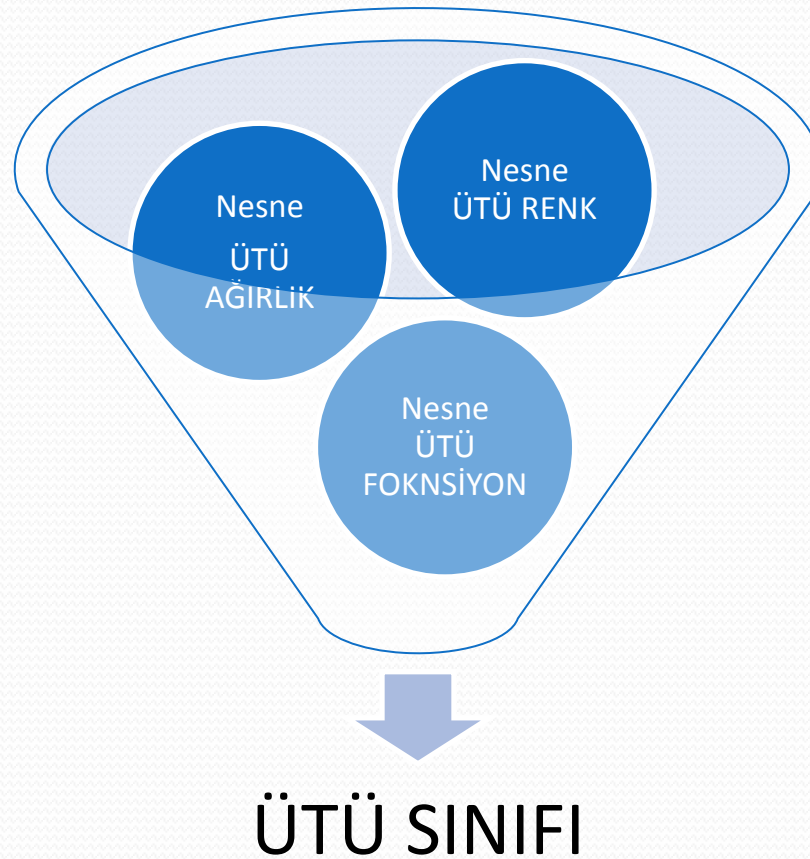
- **System.Data:** Veritabanı işlemlerinin tamamı için hazır gelen sınıf kütüphanesine bu isim alanı ile erişilir.
 - Bu sınıf kütüphanesi içindeki SQL ile işlemler için “System.Data.SqlClient” isim alanı mevcuttur.
- **System.Xml:** Veri biçimlendirme ve internetten veri paylaşımı için en çok kullanılan teknolojilerden biri olan XML ile çalışmak için gerekli sınırları içerir.

Namespaces and .NET Class Library

(İsim Alanları ve .NET Sınıf Kütüphanesi)

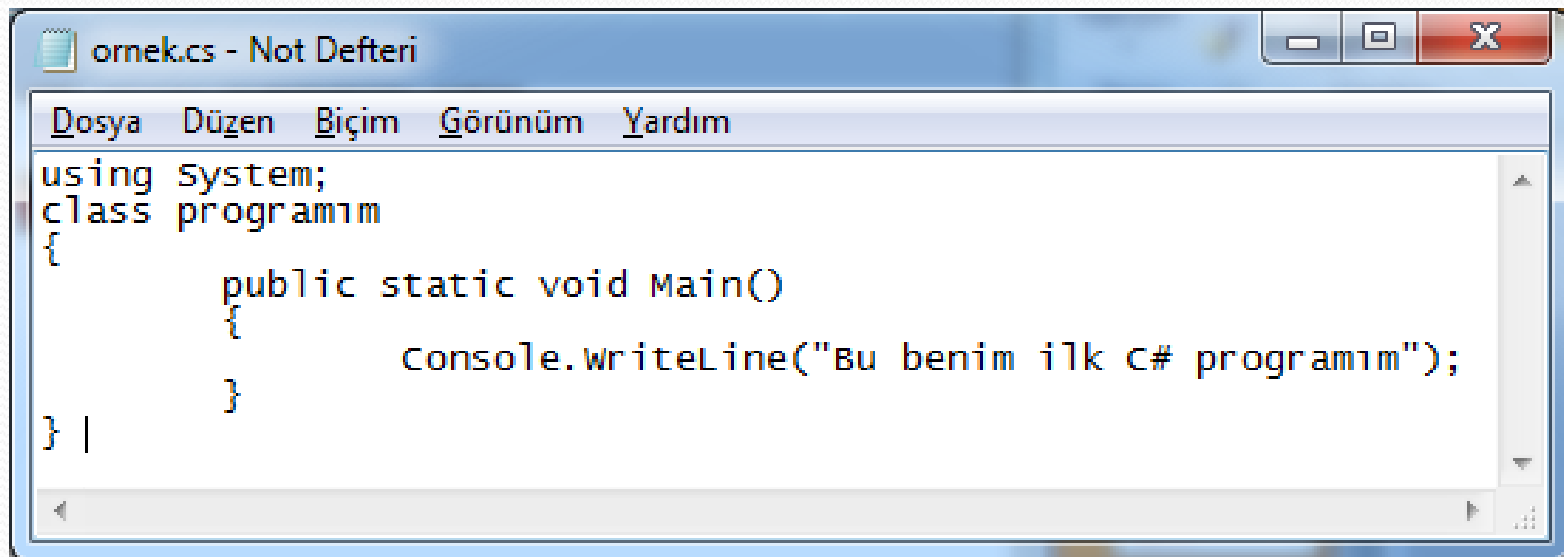
- **System.Net:** Dağıtık uygulama geliştirmek için gerekli olan ağ bileşenlerini içeren isim alanıdır. HTTP ve ağ protokolleri için kullanılır.
- **System.IO:** Dosyalarla çalışmak (okuma/yazma) için gerekli işlemlerini içerir.
- **System.Windows.Forms:** Windows temelli uygulamalarda kullanılan görsel kontrolleri barındıran isim alanıdır

Nesne Yönelimli Programlama (C#)



Bir C# programını Derlemek ve Çalıştırmak

- C# programını bir metin editörü kullanarak oluşturmalsınız.
- **1)** ornek.cs (uzantısı cs) olacak şekilde kaydediniz.

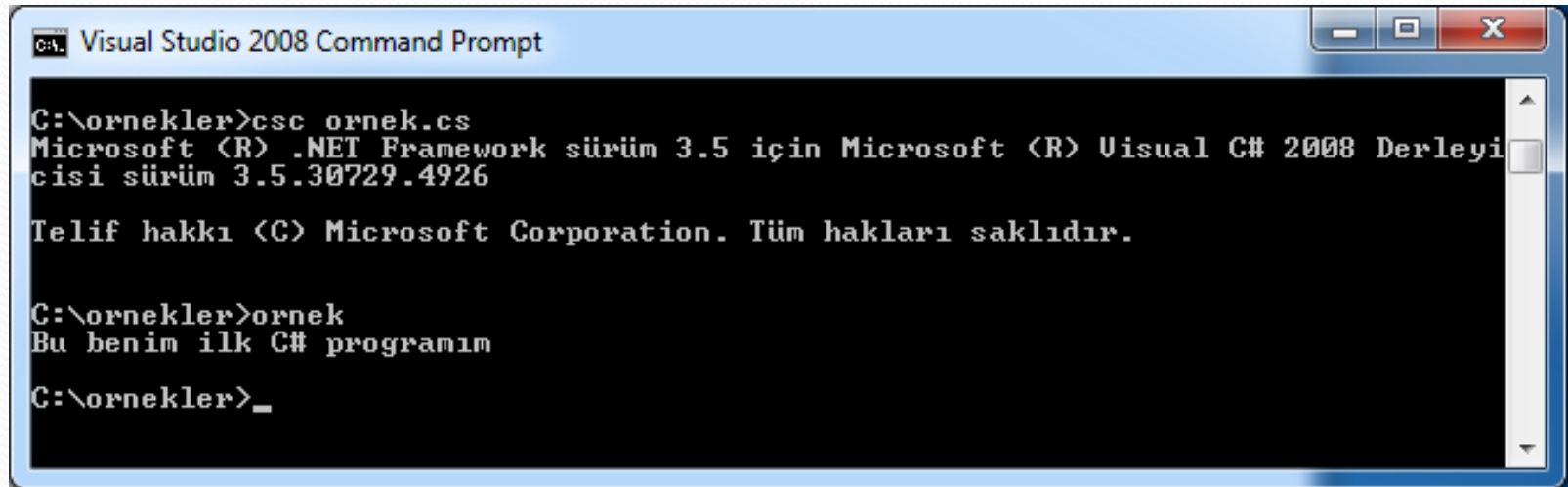


```
using System;
class programım
{
    public static void Main()
    {
        Console.WriteLine("Bu benim ilk C# programım");
    }
} |
```


Bir C# programını Derlemek ve Çalıştırmak

- **2)** Başlat/Programlar/Microsoft Visual Studio.NET 200x/ Visual Studio.NET tools/ Visual Studio.NET Command Prompt seçeneğine tıklayarak komut satırını açınız.
- **3)** C:\>csc ornek.cs komut satırını kullanarak programınızı derleyeliniz.
- **4)** Son olarak sadece programın adını yazarak çalıştırınız.
- C:\> ornek
- **Bu benim ilk C# programım**

Bir C# programını Derlemek ve Çalıştırmak



```
Visual Studio 2008 Command Prompt

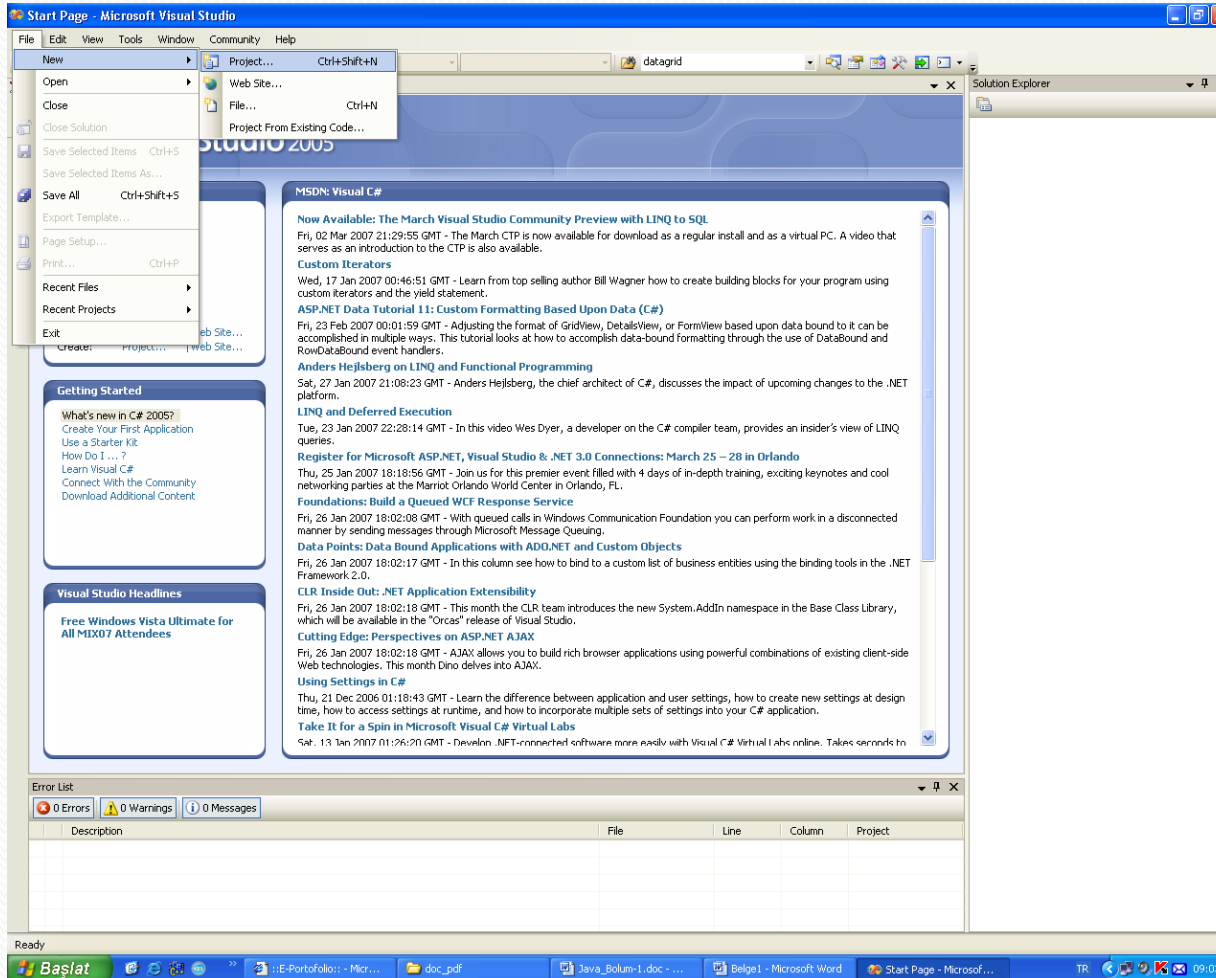
C:\ornekler>csc ornek.cs
Microsoft (R) .NET Framework sürüm 3.5 için Microsoft (R) Visual C# 2008 Derleyici
sürüm 3.5.30729.4926

Telif hakkı (C) Microsoft Corporation. Tüm hakları saklıdır.

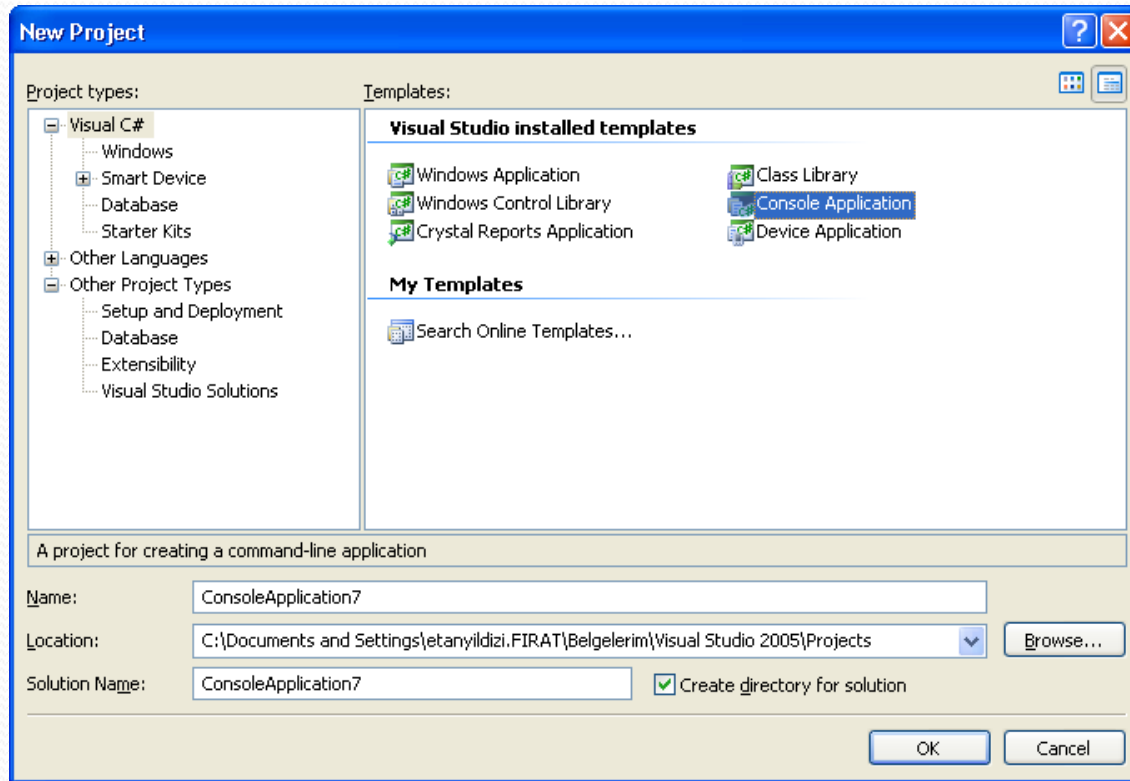
C:\ornekler>ornek
Bu benim ilk C# programım

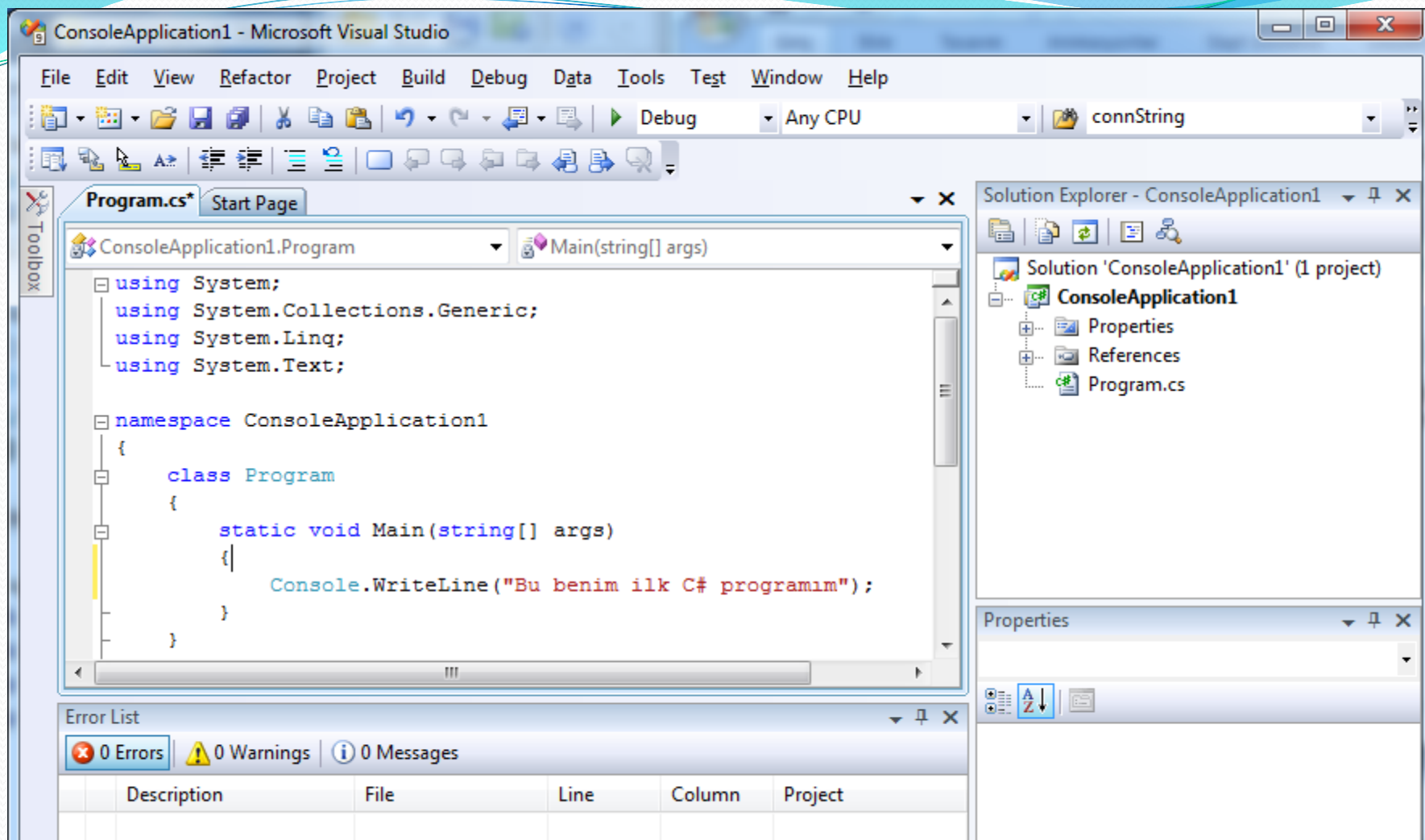
C:\ornekler>_
```

Bir C# programını Derlemek ve Çalıştırmak



Bir C# programını Derlemek ve Çalıştırmak





Önemli

- `class ilk_program1`
- `{ static void Main()`
- `{ System.Console.WriteLine("Merhaba C#"); }`
- `}`
- C# dili daha önce de denildiği gibi %100 nesne yönelimli bir dildir. Nesne olmayan hiçbir şey yoktur. C ve C++ dillerinde programın çalışması main işlevinden başlar ancak main işlevi hiçbir zaman bir sınıf içerisinde olmamıştır.
- **C# dilinde her şey sınıflarla temsil edildiği için main işlevi de bizim belirlediğimiz bir sınıfın işlevi olmak zorundadır.**

Önemli

- Bütün C# programları en az bir sınıf içermelidir. Sınıf bildirimi içinde olmayan programlar derlenmez.
- **Main()** işlevi bizim için C ve C++ dillerinde olduğu gibi programımızın başlangıç noktasıdır
- **C#**'de diğer bazı dillerde olduğu gibi kaynak koddaki bütün satırlar “**;**” ile sonlandırılır. (Bazı durumlar hariç.)

Önemli

- Sınıflar ve işlevler açılan ve kapanan küme parantezler **{ }** içerisine yazılırlar.
- C# dilinde birçok kavram sınıf dediğimiz nesneler üzerine kurulmuştur. Her sınıfın iş yapan çeşitli elemanları vardır. İş yapan bu elemanlara **metot** ya da **işlev** denilmektedir.

Önemli

- .NET' i meydana getiren sınıf kütüphanesi hiyerarşik bir yapı sunmaktadır. Sınıflar **isim alanı (namespace)** dediğimiz kavramla erişilmesi kolay bir hale gelmiştir.

```
using System;

namespace Ornekan1
{
    public class Program1
    {
        static void Main(string[] args)
        {

        }
    }

    public class Program2
    {
        public void fonsiyon1()
        {

        }
    }

    public void fonsiyon2()
    {

    }
}

namespace Ornekan2
{
    public class Program3
    {

    }
}
```

Önemli

```
using System;  
class ilk_program2  
{ static void Main()  
    {  
        Console.WriteLine("Merhaba C#");  
    }  
}
```

- **“using System”** deyimi ile System isim alanındaki bütün sınıflara doğrudan erişim hakkına sahip oluruz.

Önemli

```
using System;
class ilk_program2
{ static void Main()
    {
        Console.WriteLine("Bir tusa basın...");
        Console.ReadLine();
        Console.WriteLine("Bir tusa basın...");
    }
}
```

- **ReadLine** metodu da **WriteLine** gibi kullanılır ancak metodun parantezlerine herhangi bir şey yazılmaz.

Derleyici Parametreleri

- **csc programadi.cs* Programadi.exe şeklinde bir dosya oluşturur.
- **csc /out: yeniprogramadi.exe programadi.cs* yeniprogramadi.exe şeklinde bir dosya oluşturur.
- **csc /t:library programadi.cs* Programadi.dll şeklinde bir dosya oluşturur.
- **csc /t:module programadi.cs* Programadi.netmodule isimli derlenmiş .net modülü elde edilir.
- **csc /t:winexe programadi.cs* Program konu bir Windows uygulaması ise kullanılır.

Derleyici Parametreleri

- **csc /o:+ programadi.cs* Derleyicinin optimize edilmiş sonucu çıkarması için kullanılır.
- **csc /unsafe programadi.cs* Pointer kullanılan programları derlemek için kullanılır.
- **csc /bugreport:RaporDosyası.txt programadi.cs* Derlenen programda eğer hata varsa RaporDosyası.txt dosyasına kaydeder
- **csc /help veya /?* C# derleyicisinin parametrelerini verir.

Visual Studio.Net -C#

2. HAFTA

Temel Veri Türleri