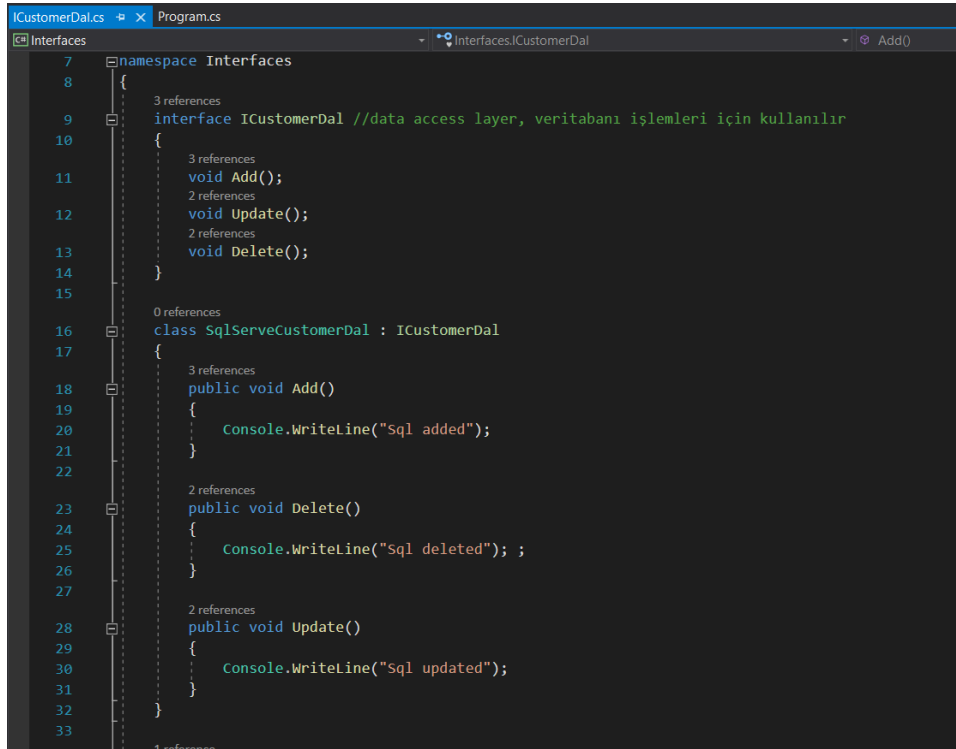


<https://www.tutorialsteacher.com/online-test/csharp-test> linki üzerinden 20 soruluk C# testini çözdüm. Bazı sorulara tereddütsüz bir şekilde doğru cevapları verdim, bazı sorularda doğru bildiğimi sandığım bilgiler yanlış çıktı, bazı sorularda da yeni kavramlarla karşılaştım. Tüm sorular için genel bir araştırma yapmaya çalıştım ama en çok yeni karşılaştığım konular üzerinde durdum. En son sayfada kaynakçayı paylaşıyor olacağım.

1- C# class can inherit multiple interfaces. (Bu soru için doğru cevap vermiş olsam bile interface ve abstract farkını daha derinlemesine görmek istediğim için araştırma yaptım)

Interface, içerisinde sadece kendisinden türeyecek olan sınıfların içeriini dolduracağı metod tanımlarının bulunduğu ve soyutlama yapmamıza olanak sağlayan bir yapıdır. Interfaceleri tanımlarken **interface** keywordünü kullanırız. Tanımladığımız yapının interface olduğunu belirtmek için isminin önüne I harfini getirmek bir best-practice olacaktır. Böylece kodlama yaparken inherit aldığımız yapının bir class mı yoksa interface mi olduğunu kolaylıkla ayırt edebiliriz. Bir interface örneği verecek olursak,



```

7 namespace Interfaces
8 {
9     3 references
10    interface ICustomerDal //data access layer, veritabanı işlemleri için kullanılır
11    {
12        3 references
13        void Add();
14        2 references
15        void Update();
16        2 references
17        void Delete();
18    }
19
20    0 references
21    class SqlServeCustomerDal : ICustomerDal
22    {
23        3 references
24        public void Add()
25        {
26            Console.WriteLine("Sql added");
27        }
28
29        2 references
30        public void Delete()
31        {
32            Console.WriteLine("Sql deleted"); ;
33        }
34
35        2 references
36        public void Update()
37        {
38            Console.WriteLine("Sql updated");
39        }
40    }
41 }
  
```

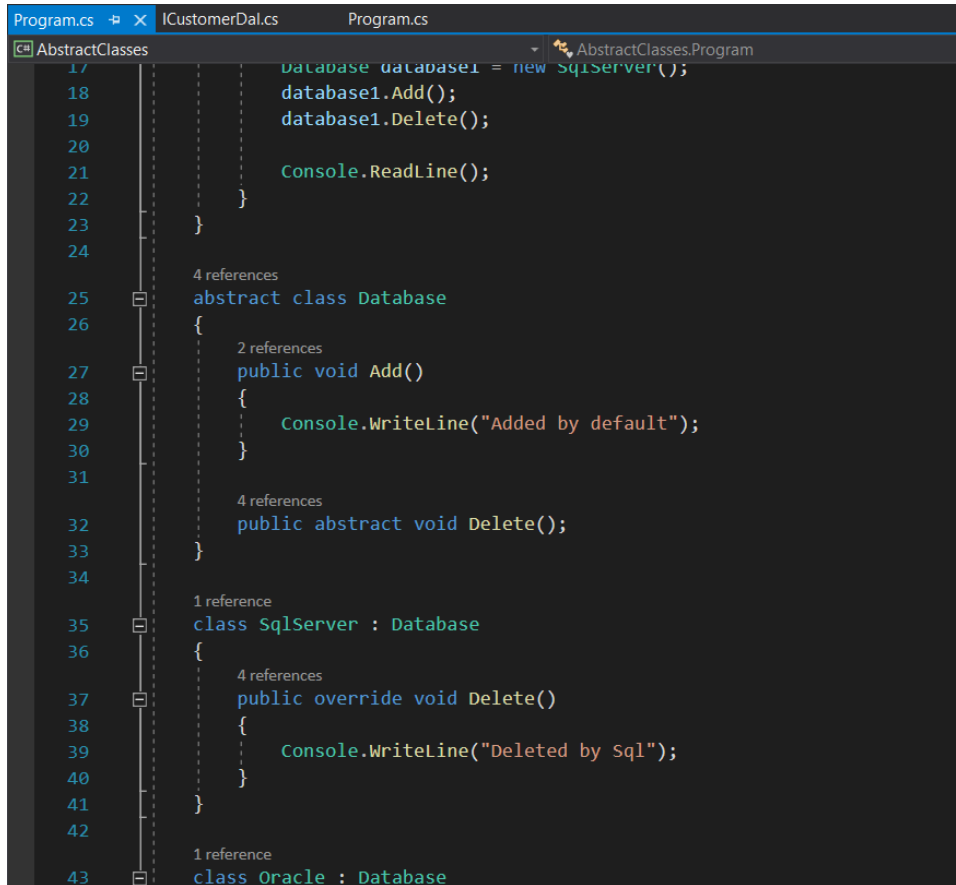
Interfaceleri bir sözleşme gibi düşünersek, türeyen tüm sınıflar void Add(), void Update(), void Delete() metodlarını implement etmek zorundadır, aksi durumda derleyici hata verecektir ve kodumuz çalışmayacaktır. Interfacelerin kısaca özelliklerinden bahsedecek olursak;

- new keywordü ile nesneleri oluşturulamaz.
- Bir sınıfın ne yapması gerektiğini belirtir, nasıl yapması gerektiğini değil.
- Default olarak tüm Interface üyeleri abstract ve public olarak tanımlanır. Özellikle belirtmeniz gerekmez.
- Bir sınıf birden fazla interface'i inherit edebilir, çoklu kalıtım (multiple-inheritance) desteklenir.

- İçerisinde yalnızca metodların imzaları yer alır, içi dolu metod bulunduramazlar.
- Kendisinden inherit alacak sınıflar ile arasında “can-do” ilişkisi vardır.

Abstract Class nedir ?

Abstract sınıflar sınıf hiyerarşisinde genellikle base class (temel sınıf) tanımlamak için kullanılan ve soyutlama yeteneği kazandıran sınıflardır. Bir sınıfı abstract yapmak için abstract keywordünü kullanırız. Abstract sınıflar en az bir tane abstract metod bulundurması bir best practice’tir.



```

17 Database database1 = new SqlServer();
18 database1.Add();
19 database1.Delete();
20 Console.ReadLine();
21 }
22 }
23 }
24 }
25 4 references
26 abstract class Database
27 {
28     2 references
29     public void Add()
30     {
31         Console.WriteLine("Added by default");
32     }
33     4 references
34     public abstract void Delete();
35 }
36 1 reference
37 class SqlServer : Database
38 {
39     4 references
40     public override void Delete()
41     {
42         Console.WriteLine("Deleted by Sql");
43     }
44 }
45 1 reference
46 class Oracle : Database

```

Özelliklerinden bahsedecek olursak;

- Abstract sınıfları genel olarak inheritance (kalıtım) uygularken kullanırız.
- new anahtar sözcüğü ile nesneleri oluşturulamaz.
- İçerisinde değişken ve metod bulundurabilir.
- Abstract sınıflardan türetilen sınıfların abstract metodları implement etmesi zorunludur. Diğer metodları override etmeden de kullanabilir.
- Constructors (yapıcı metodlar) ve destructors (yıkıcı metodlar) bulundurabilirler.
- Static tanımlanamazlar. (Tanımlanmaya çalışılırsa compiler “an abstract class cannot be sealed or static” hatası verir)
- Bir sınıf yalnızca bir abstract sınıfı inheritance yoluyla implement edebilir. Çoklu kalıtım (multiple inheritance) desteklenmez.
- Abstract olmayan metodları da bulundurabilir.
- Kendisinden inherit alacak sınıflar ile arasında “is-a” ilişkisi vardır.

Abstract ve Interface farkları için güzel bir tablo buldum. Hemen aşağıya ekliyorum.

ABSTRACT CLASS	INTERFACE
Constructor içerebilir.	Constructor içeremez.
Static üyeler içerebilir.	Static üyeler içeremez.
Farklı tiplerde access modifier (erişim belirleyicisi) içerebilir. public, private, protected gibi.	Farklı tiplerde access modifier içeremez. Interface'te tanımlanan her metod default olarak public kabul edilir.
Sınıfın ait olduğu kimliği belirtmek için kullanılır. (is-a ilişkisi)	Sınıfın yapabileceği kabiliyetleri belirtmek için kullanılır. (can-do ilişkisi)
Bir sınıf sadece bir tane abstract sınıfı inherit edebilir.	Bir sınıf birden fazla interface'i inherit edebilir.
Eğer birçok sınıf aynı türden ve ortak davranışlar sergiliyorsa abstract sınıfı base class olarak kullanmak doğru olacaktır.	Eğer birçok sınıf yalnızca ortak metodları kullanıyor ise interface'ten türetilmeleri doğru olacaktır.
Abstract sınıf metod, fields, contents vb. üyeleri içerebilir.	Interface yalnızca metod imzalarını içerebilir.
Türetilen sınıflar abstract sınıfı tamamen veya kısmi implemente edebilir.	Türetilen sınıflar interface'i tamamen implement etmek zorundadır.
Metod imzaları veya implementasyonları içerebilir.	Yalnızca metod imzalarını içerebilir.

2- Which of the followings are value types in C#? All of the above (Int32, Double, Decimal)

3- Which of the following is a reference type in C#? String

(2. ve 3. sorular hakkında genel bir bilgim olsa da daha detaylı bilgi edinmek istedim, string'i değer türü olarak bilirken referans türü olduğunu öğrendim.)

Değer Türleri ve Referans Türleri

Bir yazılımda işlenecek veriyi RAM' de tutabilmek için değişkenler kullanılır. Yazılım adına ram'e veri koymamızı ve daha sonrasında ihtiyaç durumunda ram'de ki o veriyi elde etmemizi sağlayan yapılara değişkenler adı verilir. C# programlama dilinde iki tür veri tipi mevcuttur. Bunlar değer türleri ve referans türleridir. Değer tipleri, veriyi taşıyan ve taşıdığı veriye göre bellek üzerinde yer dolduran değişken türleridir. Bellekte az yer kaplarlar ve hızlı bir şekilde erişilebilirler. Referans türleri ise, bellek bölgesinde veri yerine adresi tutarlar ve o adresin gösterdiği yerde de veri tutulur. Yani, referans türlerinde nesnenin adresi üzerinden işlem yapılmaktadır. Değer türleri belleğin stack bölgesinde tutulurken, referans türleri heap bölgesinde tutulur. Stack'te tutulan öğelerin işi bitince kendiliğinden bellekten silinirler.

Referans tiplerin işi bitince, çöp toplayıcı (garbage collector) onları toplayıp siler ve boşalan bellek bölgesini heap' e katar.

Değer türleri: int, long, float, double, decimal, char, bool, byte, short, struct, enum

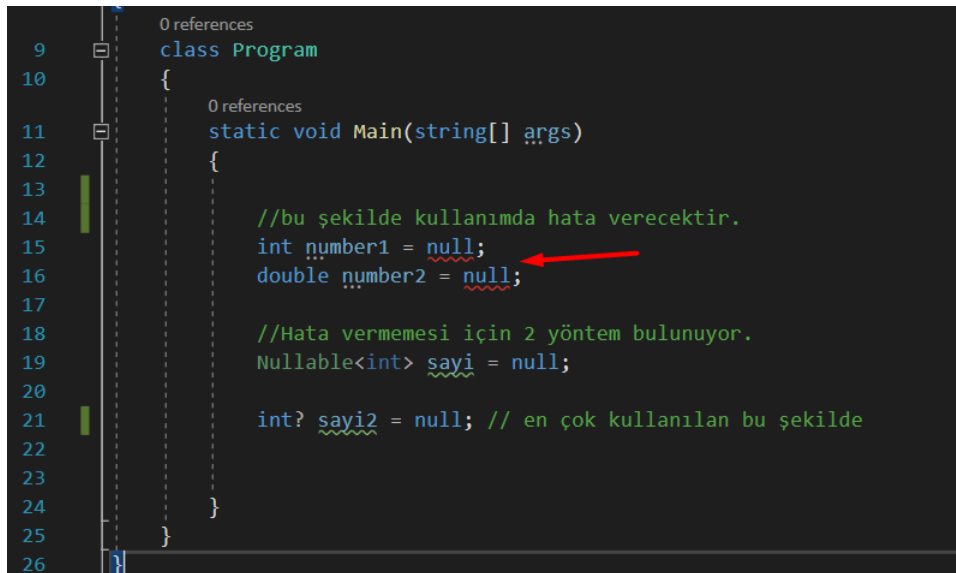
Referans türleri: string, object, class, interface, array, delegate, pointer şeklinde sıralanabilir.

4- What is Nullable type? **It allows assignment of null to value type.**

(Bu sorunun cevabını bilmiyordum, hem araştırmasını yapıp hem de visual studio üzerinden denemesini yaptım.)

Nullable Type Nedir?

C#' da içerisine boş değer (null) atanabilir değerlere nullable değer denir. Nullable tipler, ilgili tipin değer aralığına ve karakteristiğine sahip olmakla birlikte ek olarak null değer de içerebilen yapılardır. Basit olarak, değişkenin değer içerip içermediği bilgisini saklar. Sadece değer tipleri (value type) nullable olabilir.



```

9 | 0 references
10 | class Program
11 | {
12 |     0 references
13 |     static void Main(string[] args)
14 |     {
15 |         //bu şekilde kullanımda hata verecektir.
16 |         int number1 = null;
17 |         double number2 = null;
18 |
19 |         //Hata vermemesi için 2 yöntem bulunuyor.
20 |         Nullable<int> sayi = null;
21 |
22 |         int? sayi2 = null; // en çok kullanılan bu şekilde
23 |
24 |     }
25 | }
26 |

```

5- Struct is a value type. (Bootcamp öncesi daha çok javayla ilgilendiğim için struct hakkında pek bir bilgim yoktu. Java'da olmayan bir yapı. Bu yüzden detaylı araştırmasını yapıp, kod üzerinde denemesini yaptım.)

Struct (Yapı) Nedir ?

Birbirleriyle ilişkili değişkenlerin, bir isim altında toplanmasına yapı adı verilir. Yapılar, değişik veri tiplerinde elemanlar içerebilirler ve dosya içinde tutulacak kayıtları oluşturmakta kullanılırlar.

Yapılar, diğer tipte nesneler kullanılarak oluşturulan, türetilmiş veri tipleridir. Örnek bir yapı şu şekilde oluşturulabilir:

```

0 references
struct ogrenci
{
    public char[] ad = new char[20];
    public char[] soyad = new char[10];
    public long no;
    public short sinif;
}

```

struct anahtar kelimesi yapı tanımını başlatır. ogrenci tanıtıcısı yapı etiketidir (structure tag). Yapı tanımında parantezler içinde bildirilen değişkenler ise yapı elemanlarıdır.

Yapı ile Sınıf Arasındaki Farklar: Yapılar, sınıflar ile büyük benzerlik gösterirler. Sınıflar gibi tanımlanırlar ve sınıflar gibi, özellikler, metotlar, veriler, yapıcılar vb. içerebilirler. Bu benzerliklere karşın yapılar ile sınıflar arasında çok önemli farklılıklar da vardır.

- Yapılar bir değer türü, sınıflar ise bir referans türüdür.
- Yapılar için varsayılan bir yapıcı metot (default constructor) yazılmak istendiğinde derleyici hatası alınır. Ancak bu, değişik sayıda parametreler alan yapıcılar yazmamızı engellemez. Sınıflarda ise, sınıfın varsayılan yapıcı metodu yazılabilir.
- Bir sınıf oluşturulduğunda, bu, başka bir temel sınıftan kalıtım yolu ile türetilebilir ancak bir yapı başka bir yapı temel alınarak türetilemez.

Yapıların kullanımı, verimlilik ve performans açısından önemlidir. Yapılar, bir değer tipi oldukları için, bir referans aracılığıyla değil de doğrudan kendi üzerlerinde işlem görürler. Böylece, bir yapı, ayrı bir referans değişkeni gerektirmez. Bu, bazı durumlarda daha az bellek kullanıldığı anlamına gelir. Üstelik, bir yapı, doğrudan erişilebildiği için, sınıf nesnelerinin erişimlerine özgü performans kaybına da maruz kalmaz. Sınıflar bir referans tipidir. Bu nedenle, sınıf nesnelerine yapılan tüm erişimler bir referans aracılığıyla olmalıdır. Bu tür bir dolaylı erişim, erişimlerin her birine ek bir yük bindirir. Yapılar böyle bir yükün altına girmezler. Genel olarak, birbirleriyle bağlantılı küçük bir veri grubu saklaması istenildiğinde ve kalıtıma ihtiyaç yoksa ya da referans tiplerinin sağladığı diğer avantajlardan yararlanılmak istenmiyorsa, yapı tipini kullanmak daha uygun bir tercih olacaktır.

6- $10 > 9$? "10 is greater than 9" : "9 is greater than 10" is an example of **Ternary operator**.

7- Which of the following datatype can be used with enum? **Int**

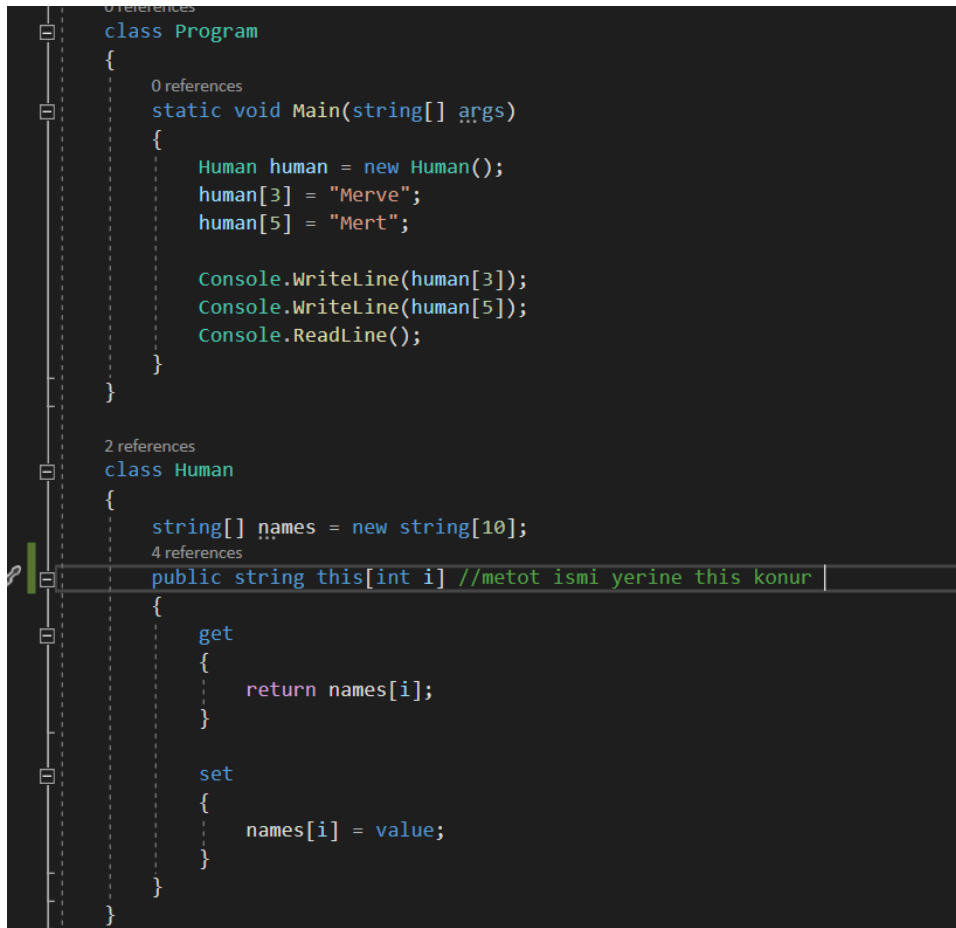
Enum nedir ?

Değişkenlerin alabileceği değerlerin sabit (belli) olduğu durumlarda programı daha okunabilir hale getirmek için kullanılır. Programda birçok değişkene tek tek sayısal değer vermek yerine "enum" kullanılabilir. Özet olarak "enum" yapısı sayıları anlamlı şekilde isimlendirerek kullanabilmeye izin verir.

8- What is indexer? **It allows an instance of a class to be indexed like an array** (Bu sorunun cevabını bilmiyordum, hatta indexer kavramını ilk kez duydum diyebilirim. Araştırmasını yapıp kod üzerinde denedim.)

Indexer nedir ?

Indexer özel tanımlı bir property'dir ve sadece class içerisinde tanımlanabilir. Tanımlandığı class'a indexlenebilir özelliği kazandırır. Array işlemlerinde kullandığımız [] operatörünü tanımlamış olduğumuz bir classı diziymiş gibi işlemler yapabilmek içinde kullanabiliriz.



```

0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        Human human = new Human();
        human[3] = "Merve";
        human[5] = "Mert";

        Console.WriteLine(human[3]);
        Console.WriteLine(human[5]);
        Console.ReadLine();
    }
}

2 references
class Human
{
    string[] names = new string[10];
    4 references
    public string this[int i] //metot ismi yerine this konur
    {
        get
        {
            return names[i];
        }

        set
        {
            names[i] = value;
        }
    }
}

```

9- String data type is **Immutable**.

10- An array in C# starts with **zero** index.

11- Which of the following is right way of declaring an array? **int[] intArray = new int[5];**

12- Which of the following is true for ReadOnly variables? **Value will be assigned at runtime.** (readonly anahtar kelimesinin tam olarak ne anlama geldiğini bilmiyordum, araştırmasını yaparken readonly ile birlikte const anlatıldığı için bu konuya da değinmiş olup aralarındaki farkı inceledim.)

Readonly nedir ?

Readonly olarak tanımlanan bir değişken yalnızca okunmak üzere tanımlanır ve yalnızca 2 şekilde değer ataması yapılabilir.

1. İlk tanımlandığında değer ataması yapılarak
2. Bir constructor içinde değer ataması yapılarak (çalışma anında(runtime) readonly bir değişkenin değeri set edilmiştir.)

Yukardaki 2 durum hariç readonly değişkenine değer ataması yapılamaz. Bu durum, aslında sadece bir yapılandırıcı tarafından değer ataması yapılması gerektiği durumlarda kullanışlıdır. Böylece daha sonra değer değiştirmesi önlenmiş olur. Ayrıca readonly ifadeler static değildir.

Const Nedir ?

Const' da aslında readonly'e benzer özellikler gösterir. Aralarındaki fark, const ifadesi yalnızca ilk değer atamasında değer verilir daha sonra program boyunca tekrar değiştirilemeyecektir. Ayrıca ilk değer ataması zorunludur. Ayrıca const ifadesi static bir ifadedir, fakat static ifadeler sonradan değişebiliyor iken constlar sabittir ve sadece tanımlandığı yerde bir kez değer ataması yapılır.

const ifadeler sadece primitive (int, double, short) değerler ile birlikte kullanılır. Referans bazlı tanımlarla birlikte kullanılamazlar.

13- Which of the following statement is true? **All of the above.**

14- Which of the following statement is true? **A return or break statement cannot exit a finally block.**

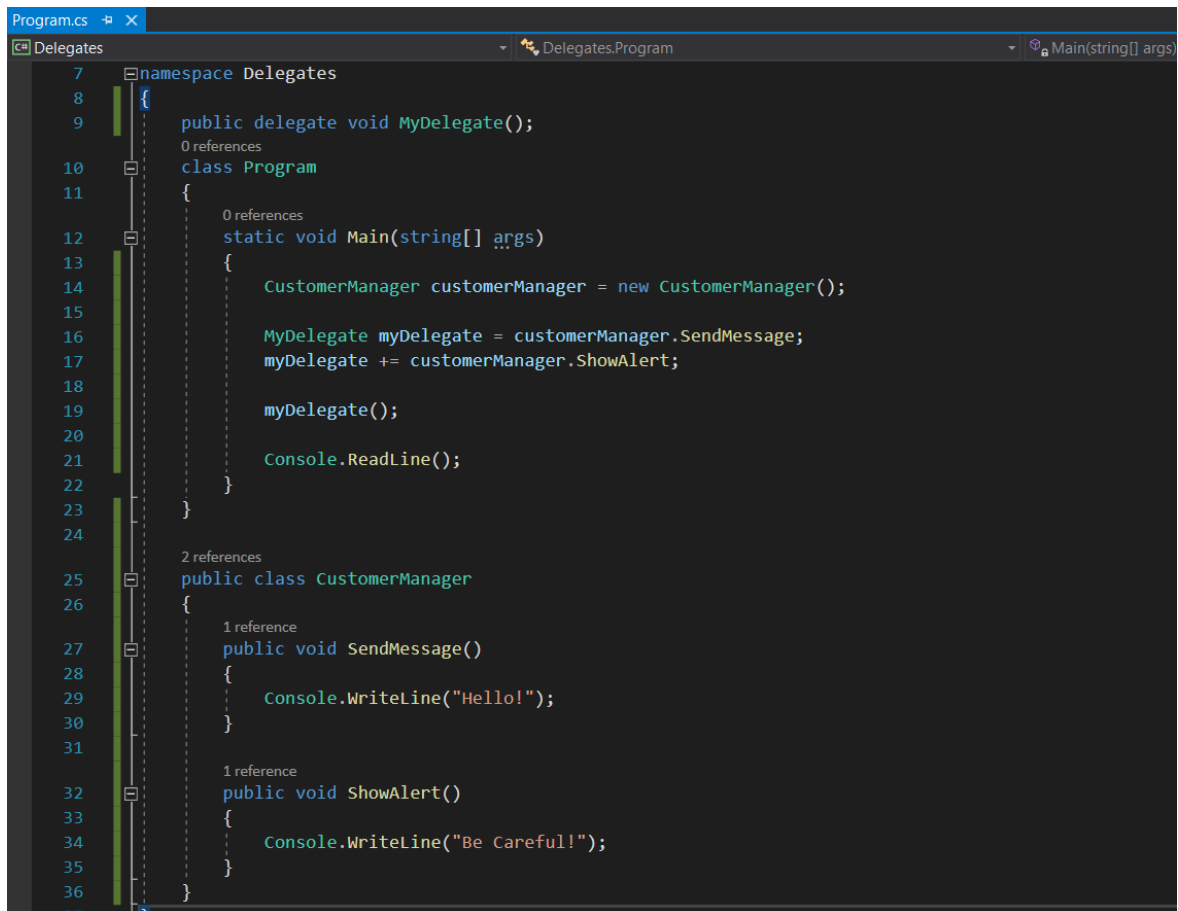
15- Func and Action are the types of **delegate.**

16- Return type of Predicate <T>() is always a **boolean.**

(15. ve 16. Sorular için pek fazla bilgim yoktu bu yüzden genel bir araştırma yapıp, yine kod üzerinde denemelerini yaptım)

Delegate Nedir ?

Delegeler, metodların adreslerini dolayısıyla metodların kendilerini tutabilen, işaret edebilen yapılardır. Delegeler referans tipli yapılardır. Yani nesne oluşturulabilirler. Bazen metodlarımızı, ihtiyacımız olduğu anda çalıştırmak isteyebiliriz. Olay(event) tabanlı programlama ve asenkron programlama yaparken, anonim metod yazarken delegelerden faydalanırız. Bir diğer kullanım amacı da, bir metoda parametre olarak başka bir metod verebilmektir. Delegeler referans türlü bir tiptir. Dolayısı ile nesneleri heap'de durur. Dikkat edilmesi gereken nokta; delegenin imzası, tuttuğu metodun imzası ile aynı olmalıdır. İmzadan kastımız, geriye dönüş tipi ve aldığı parametrelerdir. Bir delege, birden fazla metod adresi tutabilir. Bu durumda FIFO (ilk giren ilk çıkar) prensibi geçerlidir. Yani metodlar, delegeye bağlanma sırasına göre çalışırlar.



Action dönüş tipi olmayan (void), Func dönüş tipi generic olan, Predicate ise dönüş tipi bool olan özel delege yapılarıdır. 16'ya kadar parametre alabilirler. Action'lar aracılığıyla metodları parametre olarak verme olanağı sağlanır, fakat burada herhangi bir sonuç dönmeyecektir. Bunu şu şekilde açıklarsak belki daha net olacaktır. Func ile benzer işlemleri yapabiliyoruz, fakat dönüş tipi de döndürebiliyoruz. Predicate konusuna gelelim. Aslında Predicate özel bir Func delegesidir. Şöyle ki her zaman bool döndürür. Filtreleme veya select işlemlerinde yoğun olarak kullanılır.

17- A partial class allows **implementation of single class in multiple .cs files**.

(Parçalı sınıflar hakkında bilgim yoktu, genel anlamda bir araştırma yaptım.)

Partial Class(Parçalı Sınıf) Nedir ?

Parçalı sınıflar Object Oriented (Nesne Tabanlı) programlamanın temelini oluşturan sınıfların parçalanmasıyla oluşmaktadır. .NET teknolojisi de dahil olmak üzere birçok teknoloji tarafından desteklenen parçalı sınıflar;

- Büyük sınıf dosyalarını parçalama
- Mantıksal katmanlama (veritabanı ve sınıf işlemlerini ayırıştırma gibi)
- Öznitelik (attribute) ve metod ayırıştırma
- Aynı sınıf üzerinde farklı yazılımcılara çalışma imkanı sağlama gibi birçok işlemde kullanılır.

.NET, parçalı sınıfları çoğunlukla sistem ile yazılım geliştiricinin çalışma alanlarını ayırmak için kullanılır. Bu nedenle .NET sistem tarafından oluşturulan ve yazılım geliştiricinin üzerinde işlem yapabileceği sınıfları parçalı olarak tanımlamaktadır. Örneğin; ASP.NET, İnternet sayfalarını oluştururken sayfaya ait sınıfı taşıyacak iki adet kod dosyası oluşturur. Üzerlerindeki sınıf parçalı olarak tanımlanan bu dosyalardan biri sistemin diğeri ise yazılım geliştiricinin düzenlemeleri için ayrılmıştır.

Parçalı sınıflar fiziksel olarak birden çok dosyada var olabilmektedirler. Ancak bu dosyalar derlenme anında, derleyici tarafından birleştirilerek mantıksal olarak tek bir dosya haline gelmektedir. Böylece sınıf bir bütün halini alır. Bu birleştirme işlemi derleme zamanında gerçekleştirildiği için yazılım üzerine hiçbir olumsuz etki yaratmaz.

18- LINQ stands for **Language Integrated Query**

19- Data type of a variable declared using var will be assigned at **Compile time**. (var değişkeni hakkında bilgim vardı fakat sorunun cevabını bilmediğim için ekstra araştırma yapıp, diğer şıklardaki seçenekler içinde araştırma yaptım.)

Var anahtar kelimesi, değişken tanımlı yaparken tür belirtmeksizin tanımlı yapmamızı sağlamaktadır. Dinamik bir türdür. Var ile tanımlanan bir değişkene atanan ilk değer program derlendiği(compile time) anda değişkenin veri türünü belirlemektedir. Örnek olarak aşağıda bir değişken tanımlanmış ve string bir değer ataması yapılmıştır. Derleme anına kadar değişken türü belli değildir. Derlendiğinde değişken string veri türüne dönüşecektir.

```
18 var Isim = " Merve Mor";
```

Compile Time ve Runtime Nedir ?

C# tarafından baktığımızda uygulama içerisinde sabit değer tanımlamak için kullanabileceğimiz 2 yöntem bulunmaktadır. Bunlar,

Compile Time(Derleme Zamanı) : Compile işlemi; oldukça basit ve akılda kalıcı bir izahla “yüksek seviyeli bir dilin daha düşük seviyeli bir dile dönüştürülmesi işlemidir”. Yani biz yazılım geliştiriciler tarafından yazılmış kod, compile işleminden sonra makinelerin seviyesine yaklaşmaktadır. İşte bu işlemi gerçekleştiren mekanizmaya Compiler(Derleyici) derken, bu süreçte yaşanan tüm işlemleri zaman açısından anlatmak için de Compile Time (Derleme Zamanı) kavramını kullanmaktayız. Compile Time hatalarının en güzel tarafı .exe dosyası daha oluşturulmadan derleme işlemi sırasında hata alınmasıdır.

Run Time(Çalışma Zamanı): Programın çalıştırıldığı andan sonlandırılincaya kadar geçen süreci anlatmak için kullanılan kavramdır.

CLR (Common Language Runtime) Nedir?





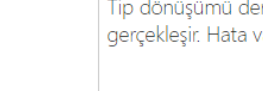
CLR (Common Language Runtime) .NET Framework’te programların çalışmasını kontrol eden ve işletim sistemi ile program arasında yer alan bir arabirimdir. CIL kodlarını çalıştıran ana mekanizmadır. CIL kodları daha önce de incelediğimiz gibi makine dili değil bir ara dildir. Bu sebeple programın çalıştırılabilmesi için bu ara dili alıp makine diline çevirecek ve makinede işletecek bir mekanizmaya ihtiyaç duymaktaydık. Bu ihtiyacı CLR karşılamaktadır. CLR, içerisinde yer alan JIT (Just in Time) derleyicisi ile, CIL kodunu alır makine koduna çevirir ve programın çalışmasını sağlar.

20- Which of the following is true for dynamic type in C#? **It escapes compile time type checking.** (Yine daha öncesinde duyduğum fakat üzerine düşmediğim bir anahtar kelime olan dynamic hakkında araştırma yaptım, aynı zamanda object tipten farkları anlatıldığı için o konuyada değindim.)

dynamic ile object arasındaki en temel fark, derleme zamanındaki tip uyumsuzluğu kontrolüdür. Bildiğiniz üzere object tipini kullanabilmemiz için unboxing (kutudan çıkarma) yapmamız gerekiyor. Yani bir tip object olarak tanımlanmışsa, karşı tarafta bunun tipinin ne olacağını belirtmeliyiz. Fakat dynamic tipi için böyle bir işleme gerek yok. dynamic tipler derleme sırasında hata almazken, object tipler için ise unboxing yapmalı, tipini açıkça belirtmeliyiz. Kısaca dynamic tipler bizi derleme hataları zahmetinden kurtarmaktadır. Fakat dynamic tipler her zaman güvenli bir program sunmayabilirler. Dikkatli kullanılmadığında çalışma zamanı hatalarına yol açabilirler.

var ile dynamic Tipler Arasındaki Farklar
dynamic tipler null olabiliyorken var tipleri mutlaka bir başlangıç değerine sahip olmalıdır. Ayrıca ilk atanan tip neyse program sonlanana kadar aynı tipte olmak zorundadır.

Object, dynamic ve var anahtar kelimelerinin farkları için güzel bir tablo buldum.

object	dynamic	var
object .NET Framework içindeki herhangi bir tipte bir değişken olabilir.	Herhangi bir tipte bir değişken olabilir	Herhangi bir tipte bir değişken olabilir fakat bir başlangıç değerine sahip olmalıdır. <pre>static void Main(string[] args) { var a; // Implicitly-typed local variables must be initialized var b = 10; }</pre>
Derleyici nesne hakkında bilgi sahibidir. 	Derleyici nesne hakkında bilgi sahibi değildir. Intellisense ile " " dan sonra herhangi bir seçenek çıkmaz. <pre>public static void dynamicTest() { int a = 10; dynamic dyn = a; dyn; // (dynamic expression) // This operation will be resolved at runtime. }</pre>	Derleyici nesne hakkında bilgi sahibidir. nesne hangi tipteysen ona göre intellisense ile bilgi edinmek mümkündür. 
Bir metoda parametre olarak gönderilebilir veya bir metodun dönüş tipi olabilir.	Bir metoda parametre olarak gönderilebilir veya bir metodun dönüş tipi olabilir.	Bir metoda parametre olarak geçilemez ya da bir metodun dönüş tipi olamaz. <pre>public static void Main() { var a = 10; var b = 20; Method(a, b); } private static void Method(int a, int b) { Console.WriteLine(a + b); }</pre>
Tip ataması yapıldığında boxing - unboxing yapılmak zorundadır.	Tip dönüşümü (boxing - unboxing) gerekmemektedir.	Tip dönüşümü (boxing - unboxing) gerekmemektedir.
unboxing sırasında tip dönüştürülemezse hata fırlatır. 	Tip dönüşümü hataları derleme zamanında anlaşılmaz ancak çalışma zamanında tip dönüştürülemezse hata fırlatır. 	Tip dönüşümü derleme sırasında gerçekleşir. Hata varsa derleme yapılır. 

Kaynakça

- <https://bidb.itu.edu.tr/sevir-defteri/blog/2013/09/06/c-'-ta-de%C4%9Fer-ve-referans-tipleri>
- <https://medium.com/software-development-turkey/bir-exe-uygulamas%C4%B1n%C4%B1n-anatomisi-clr-cil-jit-cts-kavramlar%C4%B1-eb5ee74a8525>
- <https://medium.com/software-development-turkey/abstract-class-ve-interface-aras%C4%B1daki-farklar-nelerdir-3c0a4f956eba>
- [https://bidb.itu.edu.tr/sevir-defteri/blog/2013/09/06/yap%C4%B1-\(struct\)-kavram%C4%B1-ve-yap%C4%B1-ile-s%C4%B1n%C4%B1f-\(class\)-aras%C4%B1daki-farklar](https://bidb.itu.edu.tr/sevir-defteri/blog/2013/09/06/yap%C4%B1-(struct)-kavram%C4%B1-ve-yap%C4%B1-ile-s%C4%B1n%C4%B1f-(class)-aras%C4%B1daki-farklar)
- <https://bidb.itu.edu.tr/sevir-defteri/blog/2013/09/06/c-dilinde-enum-yap%C4%B1s%C4%B1#:~:text=Bu%20yap%C4%B1%20yaz%C4%B1l%C4%B1m%20dilinde%20enum,vermek%20yerine%20%22enum%22%20kullan%C4%B1labilir>
- <https://medium.com/software-development-turkey/kar%C4%B1%C5%9Ft%C4%B1r%C4%B1lan-3-ifade-const-static-ve-readonly-40e5e2a62b62>
- <http://letsgotocoding.blogspot.com/2018/02/cta-readonly-ve-const-ifadeleri.html>
- <https://www.hikmetokumus.com/makale/29/csharp-ile-readonly-kullanimi#:~:text=Readonly%20tan%C4%B1ml%C4%B1%20de%C4%9Fi%C5%9Fkeni%20salt%20okunur,s%C4%B1n%C4%B1f%C4%B1n%20constructor%20metodu%20i%C3%A7erisinde%20yap%C4%B1lmaktadır>
- <http://onursalkaya.blogspot.com/2011/03/c-delegate-nedir.html>
- <https://medium.com/software-development-turkey/action-func-predicate-961c2d7f9bfd>
- [https://bidb.itu.edu.tr/sevir-defteri/blog/2013/09/08/partial-class-\(par%C3%A7al%C4%B1-s%C4%B1n%C4%B1f\)](https://bidb.itu.edu.tr/sevir-defteri/blog/2013/09/08/partial-class-(par%C3%A7al%C4%B1-s%C4%B1n%C4%B1f))
- <http://letsgotocoding.blogspot.com/2018/02/cta-bos-braklabilen-tipler-nullable.html>
- [https://enginpolat.com/nullable-tipler/#:~:text=Nullable%20tipler%2C%20ilgili%20tip'in,de%C4%9Fer%20i%C3%A7erip%20i%C3%A7ermedi%C4%9Fi%20bilgisini%20saklar.&text=Unutmay%C4%B1n!,\(value%20type\)%20nullable%20olabilir](https://enginpolat.com/nullable-tipler/#:~:text=Nullable%20tipler%2C%20ilgili%20tip'in,de%C4%9Fer%20i%C3%A7erip%20i%C3%A7ermedi%C4%9Fi%20bilgisini%20saklar.&text=Unutmay%C4%B1n!,(value%20type)%20nullable%20olabilir)
- <https://bidb.itu.edu.tr/sevir-defteri/blog/2013/09/06/c-dilinde-enum-yap%C4%B1s%C4%B1>
- <https://www.hikmetokumus.com/makale/40/csharp-ile-var-anahtar-kullanimi#:~:text=Var%20anahtar%C4%B1%2C%20de%C4%9Fi%C5%9Fkeni%20tan%C4%B1ml%C4%B1%20yaparken,string%20bir%20de%C4%9Fer%20atamas%C4%B1%20yap%C4%B1lm%C4%B1%C5%9Ft%C4%B1r>
- <http://letsgotocoding.blogspot.com/2018/02/cta-dynamic-tipler.html>