* Kaç tip Access Modifier vardır, özellikleri nelerdir ?
* Factory, Singleton, MVC Design Pattern’ larını açıklar mısın ?
* Compiler ve Interpreter arasındaki fark nedir ?
* Polymorphism nedir ? Neden kullanılırız ?
* Interface nedir, neden kullanılırız ?
* Abstract class, fonksiyon nedir ?
* Interface ile Abstract class arasındaki fark nedir ?
* Reflection nedir ?
* Deadlock nedir, açıklar mısın ?
* Thread çakışması nasıl önlenir ? Java da bunu nasıl yaparsın ?
* Java da Thread ve Runnable arasındaki fark nedir ?
* JVM nasıl çalışır ?
* Try-catch nedir ? Neden kullanılır ?
* Bildiğin Data Structure’ lar neler, açıklar mısın ?
* List ile ArrayList arasındaki fark nedir ?
* Set veri yapısını açıklar mısın ?
* Set veri yapısında search yapmanın time complexity’ si nedir ?
* Hash table, hash code nedir ? Neden kullanırız ?
* Java da final ve finally keyword’ leri arasındaki fark nedir ? Finally kullanımına örnek verir misin ?
* Sql de join nedir, deadlock nedir ? Left join- Right join arasındaki fark nedir ?
* Primary Key, Foreign key nedir, açıklar mısın ?
* Web servis nedir ?
* İletişim protokolleri nelerdir, açıklar mısın ?
* Http methodları nelerdir, açıklar mısın ?
* Front-end ile back-end nasıl çalışır ve haberleşir açıklar mısın ?
* JavaScript compile edilir mi ?
* Dependency Injection nedir, nasıl yapılır ? (Spring sorusu)
* Bean nedir, açıklar mısın ?
* Hibernate nedir, JPA nedir, açıklar mısın ?
* Jdbc bağlantısını nasıl kurarsın ?
* Kullandığın diller arasındaki performansı en iyi olandan başlayarak sıralayabilir misin ?
* Sana bir proje verildiğinde hangi dili kullanacağına nasıl karar verirsin ?
* Neden Java ? Java ile hangi problemleri daha iyi çözdün ?

<https://tr.myservername.com/top-25-web-developer-interview-questions>

<https://www.muratoner.net/net-developer-mulakat-sorulari-ve-cevaplari>

**Soru:** Nesneye yönelik programlamanın temel kavramları nelerdir?

**Cevap:**Maddeler halinde açıklamak gerekirse;

* Nesne (Object); Kendine has özellikleri olan ve özelliklere bağlı olarak belirli bir durumda bulunarak çeşitli tutumlar sergileyen somut ya da soyut varlıklardır.
* Sınıf (Class); Aynı türden nesnelerin oluşturduğu bir kümedir.
* Metot; Bir nesnenin yapabileceklerini belirleyen kavramdır.
* Kalıtım (Inheritance); Bir sınıfın belirli özellik ve tutumlarını bir üst sınıftan alarak, kendisi için farklı olan özellik ve tutumları ayrıca uyguluyor olmasıdır.
* Soyutlama (Abstraction); Nesnenin uygulama için gerekli olan özelliklerinin kodlanarak, diğer özelliklerinin kodlanmaması anlamına gelir.
* Çok biçimlilik (Polymorphism); Nesnelerin kendi türlerinin dışında farklı türler olarak da temsil edilmelerini sağlar.
* Kapsülleme (Encapsulation); Nesnenin yaptığı işler ve durumuyla ilgili bilgilerin ve işlevsel karmaşıklığının gizlenmesi anlamına gelir.

**Soru:** SOLID prensipleri nelerdir?

**Cevap:**Nesneye yönelik programlamadaki standartlardır. Detaylı açıklamak gerekirse;

* Single Responsibility: Bir nesnenin sadece bir sorumluluğu olmalıdır.
* Open Closed: Nesne genişlemeye açık, değişikliğe kapalı olmalıdır.
* Liskov's Substitution: Nesneler, programın çalışmasında sorun yaratmadan kendi alt örnekleriyle değiştirilebilmelidir.
* Interface Segregation: Nesneler ihtiyaç duymadıkları metotların Interface'lerinden ayrıştırılmalıdır.
* Dependency Inversion: Yüksek seviyeli sınıflar, düşük seviyeli sınıflara bağlı olmamalı, her ikisi de soyut kavramlara bağlı olmalıdır.

**Soru:** Library (Kütüphane) nedir?

**Cevap:**Önceden yazılmış hazır metotların olduğu alandır. Projeye dahil edilerek kullanılır.

**Soru:** Framework (Çatı) nedir?

**Cevap:**Önceden hazırlanmış kütüphanelerin bulunduğu bir iskelettir. Proje o iskelete göre inşa edilebilir.

**Soru:** Interface nedir?

**Cevap:**İçinde sadece kendisinden türeyen sınıfların içini doldurmak zorunda olduğu içi boş metot tanımlarının yapıldığı bir yapıdır. Bir nevi sözleşme gibidir.

**Soru:** Abstract nedir?

**Cevap:**İçinde içi dolu metotların, değişkenlerin ve boş metotların tanımlanabildiği yapılardır. Ortak özellikleri bulunan nesneleri tek bir çatı altında toplayıp kod tekrarını azaltmak için kullanılır.

**Soru:**Compiler (Derleyici) nedir?

**Cevap:**Herhangi bir programlama dilindeki kaynak kodları makine diline çeviren uygulamadır.

**Soru:**HTTP nedir?

**Cevap:**İnternette sunucular ve son kullanıcılar arasında bilgilerin nasıl aktarılacağına dair kuralları ve yöntemleri düzenleyen sistemdir.

**Soru:**Authentication nedir?

**Cevap:**Kullanıcının kimliğini kontrol etmektir. Sisteme giriş yapmak için kullanılır.

**Soru:**Authorization nedir?

**Cevap:**Kullanıcının kaynaklara erişimini doğrulamaktır. Sistemde işlem yapmak için kullanılır.

**Soru:**MVC ne anlama gelir?

**Cevap:**Model, View ve Controller kelimelerinin kısaltmasıdır.

* Model; Verileri temsil eden kısımdır.
* View; Kullanıcının gördüğü arayüzdür.
* Controller; Projedeki veritabanı işlemleri, hesaplamalar gibi her şeyin kontrol edildiği bölümdür.

**Soru:** API nedir?

**Cevap:** Herhangi bir uygulamanın belli işlevlerini diğer uygulamalarında kullanabilmesi için oluşturulan modüldür.

**Soru:**SOAP nedir?

**Cevap:**Simple Object Access Protocol'ün kısaltmasıdır. Türkçe olarak Basit Nesne Erişim Protokolü anlamındadır. İnternet üzerinden küçük miktarda bilgileri aktarma protokolüdür. Veriler genelde HTTP, bazen de TCP/IP ile gönderilir. XML formatındadır. XML kullanmaya mecbur bırakır.

**Soru:**REST nedir?

**Cevap:**İstemci ve sunucu arasında veri alışverişinin basit bir yoludur. JSON, XML, Text gibi formatlarda veri alışverişi yapılabilir. HTTP ile gönderilir. SOAP'a göre daha hızlı çalışır.

**Soru:**Middleware nedir?

**Cevap:**Ara katmandır. Örneğin bir Rest Api tasarlıyoruz. Token ile authentication işlemlerini hallediyoruz. Token gerektiren birkaç tane Api yazdık. Bu Api'lere gelen isteklerde Token'ın geçerliliğini kontrol etmek adına bir Middleware oluşturabiliriz. Böylece gelen istek Middleware'daki kontrolden geçemezse direkt olarak kesilir. Geçerse yoluna devam eder.

**Soru:**Lazy Loading nedir?

**Cevap:**Bir kod bloğu, nesne veya program parçasının sadece ihtiyaç duyulunca çalıştırılmasıdır.

**Soru:**SSL nedir?

**Cevap:**Sunucu ve istemci arasında verilerin şifrelenerek gönderilmesidir.

**Soru:**FTP nedir?

**Cevap:**Bir bilgisayardan başka bir bilgisayar ya da sunucu arasında bağlantı kurup dosya paylaşma protokolüdür.

**Soru:**Access Modifiers nedir?

**Cevap:**Sınıflara, nesnelere, metotlara vs. proje içerisinde nerelerden erişilebiliceğini tanımlayan kavramlardır.

* Public; Proje içerisinde herhangi bir yerden erişilebilir.
* Protected: Bulunduğu sınıf ve ondan türetilen diğer sınıflardan erişilebilir.
* Private: Sadece bulunduğu sınıf üzerinden erişilebilr.

**Soru:**Coupling ve Cohesion nedir?

**Cevap:**Coupling; Sınıfların birbiriyle olan bağımlılığıdır. İyi bir yazılım için Coupling'in en aza indirilmesi önerilir. Cohesion; Bir sınıfın oluşturulma amacını ne kadar temsil ettiğidir. İyi bir yazılım için Cohesion'ın yüksek oranda olması önerilir. Örneğin matematik işlemi yapan bir sınıfta dosya işlemi olmamalıdır.

**Soru:**SQL'de bulunan Index, Trigger ve Stored Procedure kavramları nelerdir?

**Cevap:**Index; İlişkili tablonun indexleme alanı olarak kullanılan kolondaki verilere göre sıralanmış biçimde işleme konulmasını sağlar. Sorguların geç cevap vermesi durumunda kullanılır. Trigger; Tetikleyici anlamındadır. Veritabanında bir işlem yapıldığında başka bir işleminde otomatik olarak yapılması için kullanılır. Stored Procedure; Belli bir işi yapan kod parçasıdır. İlk derlemeden sonra bir daha derlenmeye ihtiyaç duymaz.

**Soru:**SQL'de Deadlock ne anlama gelir?

**Cevap:**Veritabanına iki ayrı işlem gönderilir ancak ikisi de diğer işlemin sonucuna bağlı olarak görevini yapacaktır. Bu ölümcül kilitlenme durumu önceden tespit edilip birine öncelik atanmalıdır.

<https://tr.bitdegree.org/tutorial/net-mulakat-sorulari/>

**Soru:**const, readonly ve static kavramlarını açıklar mısınız?

const: Bir değişken tanımlaması sağlanır, fakat tanımlanan bu değişken daha sonra hiçbir şekilde değiştirilemez.

static: Metod, değişken veya sınıf tanımlanması için kullanılabilir, fakat dikkat edilmesi gereken şey static sınıf içine yalnızca static metod yazılır, static metod içine ise static değişkenler tanımlanmak zorundadır.

readonly: değişken ile birlikte kullanılır. const ile farkı sadece tanımlandığı yerde değil, ayrıca constructor içinde de tanımlanabilir.  [[Dahası](https://medium.com/software-development-turkey/kar%C4%B1%C5%9Ft%C4%B1r%C4%B1lan-3-ifade-const-static-ve-readonly-40e5e2a62b62)]

**Soru:**IQueryable ile  IEnumurable farkları nelerdir?

IQueryable ile IEnumerable arasında en önemli fark, IQueryable sorgu düzenlenip sunucu tarafında çağrılırken, IEnumerable ise çekilen verinin tamamını belleğe alıp oradan istenildiği şekilde düzenlenebilir. [[Dahası](https://medium.com/software-development-turkey/iqueryable-ienumerable-d52433b708da)]

**Soru:**lazy loading - eager loading nedir?

Linq’nun veri tabanından veri çekerken kullandığı iki yöntemden biridir. Birbirine bağlı tablolarda tüm veriyi çekmek yerine istemci talep ettikçe verinin gelmesidir. Diğer yöntem olan eager loading ise tüm veriyi belleğe yükler. [[Dahası](https://medium.com/software-development-turkey/iqueryable-ienumerable-d52433b708da)]

**Soru:**deffered execution - immediate execution nedir?

Deferred Execution ertelenmiş çalışma demektir. Şöyle ki linq sorguları hemen çalışmaz. Belirli bir komuta kadar bekler. Bunu tetikleyenlerden bir tanesi de ToList() metodudur. Sorgu düzenlenir, fakat çalıştırılmaz ta ki ToList() metodu çağrılır ve veri tabanına istek yapılıp veri getirilir. “Immediate Execution” ise sorgunun hemen çalıştırılmasıdır. [[Dahası](https://medium.com/software-development-turkey/iqueryable-ienumerable-d52433b708da)]

**Soru:**yield keyword'ü nedir?

Iterator işlemlerde genellikle IEnumurable ile kullanılıp bunu doldurabiliyoruz. return ve break ile olmak üzere 2 kullanım şekli vardır. [[Dahası](http://safakunel.blogspot.com/2013/12/c-yield-sozcugu-kullanimi.html)]

**Soru:**action, funtion ve predict kavramlarını açıklar mısınız?

Aralarında en temel farkı kısaca açıklayacak olursa; Action dönüş tipi olmayan (void), Func dönüş tipi generic olan, predicate ise dönüş tipi bool olan özel [delege](https://www.buraksenyurt.com/post/C-2-0-ile-Generic-Delegates-bsenyurt-com-dan)yapılarıdır. 16'ya kadar parametre alabilirler. Ufak hilelerle bunu daha da artırabiliyoruz.[[Dahası](https://medium.com/software-development-turkey/action-func-predicate-961c2d7f9bfd)]

**Soru:**implicit ve explicit hiç duydunuz mu? (Operatörlara aşırı yükleme)

Operatörlere aşırı yükleme diye Türkçeye çevrilen kavram metodlara aşırı yükleme ile benzer mantıkta olsa da farklı işlemlerdir. Genel kullanımı +, - gibi operatörlere aşırı yükleme yapılır. Normal koşullarda 2 nesne toplanamaz, fakat bu operatörlere aşırı yükleme yaparak artık bu operatörleri istediğiniz gibi kullanabiliyorsunuz. Şöyle ki; implicit ve explicit diye 2 tür dönüştürme vardır. Implicit aritmetik işlemleri içerir ve veri kaybı olmadığından cast işlemi gerektirmez. Explicit ise cast işlemi gerektirir, veri kaybı olma ihtimali de vardır. [[Dahası](https://medium.com/software-development-turkey/operat%C3%B6rlere-a%C5%9F%C4%B1r%C4%B1-y%C3%BCkleme-implicit-explicit-9f8df5a248af)]

**Soru:**Covariance<out T> — Contravariance<in T> kavramlarını duydunuz mu?

Bahsedilen iki yapı da generic interface’lerle kullanılır. Fakat interface out keyword ile tanımlanırsa onu implemente eden sınıf dönüş tipi sadece T olan metodlar döndürebiliyor, in keyword’ü kullanıldığında ise T tipi sadece parametre olarak kullanılabilinmektedir. [[Dahası](https://medium.com/software-development-turkey/in-t-ve-out-t-ile-generic-inteface-kullan%C4%B1m%C4%B1-contravariance-covariance-ede546119cc4)]

**Soru:**var dynamic keyword'leri hakkında kısaca bilgi verebilir misiniz?

var C# 3 ile gelen ve değişkenin tipini build time’de belirten keyword’dür. Örneğin nesneyi string bir değişkene çevirdiniz. Bunu intellisense özelliği sayesinde build time’de string özelliklerini yansıtacaktır ve string metodları görüntüleyebileceğiz. dynamic ise C# 4 ile gelen ve değişkenin tipini run time’de belirten bir keyword’dür. Aynı örnek üzerinden devam edersek, nesneyi string’e dönüştürmüş dahi olsak tipinin değişebilmesi sebebiyle string metodlarını göstermeyecektir, fakat herhangi bir değişiklik yapılmayıp bu metodlar çağrılırsa doğru olarak çalışacaktır.

**Soru:**as ve classic cast () arasındaki fark nedir?

classic cast ile bahsettiğimiz şey “(ClassName) obj” tam olarak budur. Buradaki cast işleminde herhangi bir hata oluştuğunda “InvalidCastException” hatası fırlatacaktır. as ile “obj as ClassName” gibi bir cast işlemi yaptığımızda ise cast işlemi hatalı olduğunda nesneyi null olarak set etmektedir. Bunlardan biri diğerinden daha iyidir demekten ziyade kullanım alanına göre değişkenlik göstermektedir. Hatayı yakalamanın maliyeti fazla olacağından classis cast işlemi çok daha hızlı olmaktadır, fakat as ile yapılan işlemlerde de program hata fırlatmayacaktır. Bunu göz önünde bulundurmakta fayda var.

**Soru:**Tip kontrolleri (casting) hakkında bilgi verebilir misiniz?

typeof: Bir keyword olup nesne veya değerin tipini almayı sağlar, ancak genelde nesnenin tipini belirlemede kullanılır. Derlenme zamanında tipi belirler. Bir nesne oluşturarak tipin değerini direkt belirler. Parametre alır.

GetType: Bir metod olup nesne veya değerin tipini almayı sağlar, ancak genelde veri tipini belirlemede kullanılır. Çalışma zamanında tipi belirler. Çalışma zamanında kullanıldığından dolayı reflection ile çok sık kullanılır. typeof tersine ikinci bir nesne oluşturarak değerin tipini belirler. Parametre almaz.

is: Bir keyword olup genel olarak kullanıcıdan gelen bir değer veya nesnenin, başka bir nesneye ait olup olmadığını kontrol eder. bool değer döner.

as: Bir keyword olup aslında tip kontrolünden ziyade bir cast işlemi yapar. Eğer cast işlemi başarılı değilse nesneye null değeri atar. is ile arasında aşağıdaki gibi bir bağlantı vardır. [[Dahası](https://medium.com/software-development-turkey/tip-kontrol%C3%BC-type-checking-231d6bf1be4f)]

**Soru:**virtual metod nedir?

virtual method kalıtıldığı sınıflar tarafından ezilebilen(override) methodlardır. Java dilinde her method default olarak virtual olarak gelmesine karşın C#'ta ise bu durum farklıdır ve bunu belirtmemiz gerekmektedir.

**Soru:**count() ve length farkı nedir?

length dizi nesnesinin bir property'sidir ve genelde dizinin boyutunu verimli bir şekilde bulmak için kullanılır. count ise linq ile arraylist'ler ile ilgili bir durumdur ve o da boyutu geri döndürür. Ek olarak length int dönerken, LongLength long dönmektedir.

**Soru:**Erişim belirteçleri hakkında bilgi verebilir misiniz?

Internal: Kendi projesi içerisinde public, farklı bir projeden/dışarıdan çağrılmak istenildiğinde ise private özelliklerini taşır. Yani aynı Assembly (dll) üzerinde istediğiniz şekilde kullanabilirsiniz ancak dışarıdan (farklı bir projeden) çağıramazsınız.

 Proteced Internal: Bir nesne protected internal olarak tanımlandığında aynı protected gibi kendi bulunduğu sınıf üzerinde ve bu sınıfı miras alan sınıflar üzerinden çağrılabilir. Artı olarak aynı proje (assembly/dll) üzerinde olmasalar dahi, tanımlandığı sınıftan türetilmiş diğer sınıfların içinden de çağırılabilirler.

  Özel bir durum olmadıkça varsayılan olarak değişkenler ve methodlar private, sınıflar ise internal'dır.

**Soru:**==, Equals aralarındaki farklardan bahsedebilir misiniz?

equals ilgili metodun içeriğinin aynı olup olmadığını kontrol ederken, == operatörü adreslerini kontrol eder. O yüzden değerleri aynı iken bile işaret edilen adresler farklıysa size false cevabı dönebilir.

**Soru:**C# ve Java'da multiple inheritance mümkün mı değil mi?

Çoklu kalıtım bir sınıfın birden fazla atasının olma durumudur. C++'da var iken, C#, Java gibi dillerde ise bulunmaktadır. Bunu mümkün hale getirmek için interface'ler kullanılmaktadır. Bilindiği üzere bir sınıf birden fazla interface implemente edebiliyor. [[Dahası](https://stackoverflow.com/questions/178333/multiple-inheritance-in-c-sharp)]

**Soru:**Record nedir?

C#'ta bir değişkeni değişmez yapmanın birden farklı yolu vardır. [C# 9](https://medium.com/software-development-turkey/c-9-0-ile-gelen-yenilikler-bd9856eac085) ile gelen init özelliği sayesinde ilk değer atadığımız değişkenleri artık set edemiyorduk, bunu nesne ve sınıf bazında da yapabiliyorduk. Fakat sınıflardan farklı olarak immutable'dir, yani değerleri asla değişmez.[[Dahası](https://www.borakasmer.com/derinlemesine-recordlar/)]

**Soru:**Değer ve referans tipler hakkında bilgi verir misiniz?

Bilindiği üzere değişkenler bellekte tutulur. Referans bazlı olanlar(string, object) heap'ta, değer bazlı (primative'ler) olanlar ise stack'te. Bir referens bazlı değişken kopyalanırken saklanan değerler kopyalanır, fakat referans bazlıda ise adres saklanır. O yüzden referans bazlı olan 1.değişkenin değeri değiştiğinde 2.nesnenin de değerinin değiştiğini göreceksiniz, fakat değer bazlıda böyle bir şey görülmez.

**Soru:**out ve ref farkı nedir, ne gibi farkları vardır?

Bir metod veya fonksiyona parametre gönderildiğinde bu değerler referans ve değer bazlı olmasına göre orijinal değerinin değişip değişmeyeceğini etkilemektedir. Primiative(int, decimal) gibi değerlerde orijinal değerin de değişmesini istediğimiz durumlarda bu keyword'leri kullanabiliriz. ref ile kullanımda başlangıç değeri atamak zorunda iken out'da böyle bir gereklilik yok. Bu gibi kodsal farklılıklar olsa da kullanım mantıkları benzerdir. [[Dahası](http://www.canertosuner.com/post/C-Ref-Out-Kullan%C4%B1m%C4%B1)]

**Soru:**Garbage collector nedir?

Her nesne oluştuğunda bellekte kendisine yer ayrılır. Bu nesnenin  farklı sebepler dolayı kullanılma süresi dolduğunda ise bunu bellekten kaldırma işini üstlenen mekanizmadır. C ve C++ dillerinde bunu yazılımcı kendisi yapması gerekiyor iken Java ve C# gibi dillerde bu işin büyük çoğunluğu otomatik olarak halledilmektedir. [[Dahası](https://medium.com/@gokhansengun/garbage-collector-nas%C4%B1l-%C3%A7al%C4%B1%C5%9F%C4%B1r-3bdf2fb20282)]

**Soru:**managed ve unmanaged code nedir?

Managed code kullanıldıktan sonra GC tarafından sistem tarafından silinen kod yapılarıdır, unmanaged code ise GC tarafından değil manuel olarak yazılımcı tarafından ele alınması gereken kodlardır. Bunu C# dilinde using statement veya dispose aracılığıyla yapabiliyoruz. [[Dahası](https://www.gokhan-gokalp.com/kodlama-yaparken-dikkat-edilmesi-gereken-bazi-performans-ve-optimizasyon-teknikleri/)]

**Soru:**Safe/Unsafe code nedir?

Bildiğiniz üzere C#'ta pointer yoktu ve bellek yönetimi de GC tarafından otomatik olarak hallediliyordu. Farklı sebeplerden dolayı bu noktaların dışında işler yapmak zorunda kalabiliyoruz. Bu da unsafe keyword'ü ile kullanılabilir hale gelmiştir.[[Dahası](https://www.gencayyildiz.com/blog/cta-gostericipointer-unsafe-komutu-ii/)]

**Soru:**Thread, thread pool ve task kavramlarını kısaca açıklar mısınız?

Thread: Gerçek thread’i temsil eder.

Thread Pool: Thread’lerin bulunduğu havuzdur.

Task: Yapılması gereken işlemleri temsil eder. [[Dahası](https://medium.com/software-development-turkey/asenkron-programlama-notlar%C4%B1m-29780d07017f)]

**Soru:**Task’leri eş zamanlı çalıştırmanın bazı yöntemlerinden bahsedebilir misiniz?

Lock/Monitoring: Dummy proje oluşturup scope içerisinde işlem yapar.

SpinLock: Çekirdeğin başka bir işlem yapmasını önler.

ReaderWriterLock: Okuma yapılırken bekletilmez, sadece yazma işlemi yaparken bekletilir.

Interlocked: Bekletme yapmadan primative tipler üzerinde işlemler yapılır.

Mutex: Farklı işlemler arasında çalışabilir. [[Dahası](https://ilkererhalim.medium.com/c-ile-paralel-programlama-2-task-senkronizasyonu-ve-veri-payla%C5%9F%C4%B1m%C4%B1-1631162b0c1d?p=1631162b0c1d)]

**Soru:**Concurrent, Parallel, Async ve Sync birbirinden farkları nelerdir?

Bu 3 kavram birbirine benzese de hatta birbirine çok karıştılsa da temelde birbirinden farklı özellikleri içerisinde barındırıyor. Bir örnek üzerinden açıklayacak olursak; bir form uygulamasından sync bir butona tıklar ve diğer tüm işlemler beklemeye geçer. Concurrent ise bir butona tıklar ve gezme işlemini yapmaya devam eder. Parallel olan ise aynı zamanda farklı thread’lerde iş yapabilir. Async ise bir işi yaparken diğer bir işin sonucu beklemez. [[Dahası](https://www.sitepoint.com/asynchronous-programming-using-async-await-in-c/)] Özetle;

Sync: Bir iş yapar, sonra diğerlerini yapar.

Concurrent: Aynı anda birden çok iş yapar.

Parallel: Aynı anda bir kopya üzerinden birden çok iş yapar.

Async: Başka bir işe başlamak için diğer işin bitmesini beklemez.

**Soru:**Concurrent kısmını biraz daha açabilir misiniz? Buna alternatif çözümler var mıdır?

Farklı thread'lerde liste veri yapısına aynı anda insert updete ve delete işlemleri yapmak istiyoruz. Bunu thread-safe olabilmeksi concurrent yapılarını kullanıyoruz. Bunları kendimiz de lock/unlock yapılar üzerinden yönetebiliriz, fakat dilin avantajlarından da yararlanabiliriz.

ConcurrentDictionary<TKey, TValue> -> Dictionary thread safe halidir.

ConcurrentBag<T> -> Sırasız şekildeki threaf safe listedir.

ConcurrentQueue<T> -> Thread safe kuyruk yapısıdır. (FIFO - İlk giren ilk çıkar.)

ConcurrentStack<T> -> Threaf safe stack yapısıdr. (LIFO - Son giren ilk çıkar.)

Yukardaki methodların TryAdd gibi methodları vardır. Thread'lerin yerini kaybetmemesi için bu

şekilde kullanılırlar. İşlemin sonlanmasını beklediği için de performansları nispeten yavaştır.

**Soru:**race condition nedir?

Atomic operasyon olmayan işlemlerde thread'lerin sırasının karışması olayına denmektedir. Şöyle ki; thread1 value değişkenin 5 değerini ekledi ama o sırada thread2 value değişkeniyle ilgili başka bir işlem yapacak olursa thread1 tekrar işlem yapacak olduğunda value değişkenini bıraktığı gibi bulamayacaktır ve bu da bir yanlışa sebebiyet verebilir. İşte bu olaya race condtion diyoruz. [[Dahası](https://medium.com/@gokhansengun/race-condition-nedir-ve-nas%C4%B1l-%C3%B6nlenir-174e9ba3f567)]

**Soru:**Configure.await(false) nedir ?

Bir thread UI'den sorumlu ve bu UI bir thread tarafından çağrıldığında bloklanır, böylece uygulama cevap veremez hale gelir. Bu problemin yaşanmaması için kullanılır. [[Dahası](https://medium.com/bynder-tech/c-why-you-should-use-configureawait-false-in-your-library-code-d7837dce3d7f)]

**Soru:**ORM nedir kullanmamız gerekli midir?

Bunun kesin doğru bir cevabı yoktur. Kullanmanın büyük avantaj olduğu aşikardır, fakat bu asla bir zorunluluk değildir. Bilhassa büyük projelerde farklı performans ve diğer bazı zorluklara da sebep olunabileceği unutulmamalıdır. İhtiyaca göre değişebilen bir cevaptır.

**Soru:**Ne zaman EF ne zaman dapper kullanırsınız?

Daha basit ve karmaşık olmayan projelerde,kullanımı ve mapleme olayı daha rahat olması sebebiyle,

EF kullanımı daha mantıklı görünürken , daha çok trafiği olan durumlar ise Dapper kullanmak

daha mantıklı gözükebilir. Her ne kadar son dönemlerde EF Core tarafında güzel performans

geliştirmeleri olmuşsa da performans olarak Dapper halen daha iyi gözükmektedir.

Zaten Dapper de kendini "light ORM" diye nitelendirdiğini hatırlatmakta fayda var.

**Soru:**AsNoTracking nedir?

Eğer ki biz yazılım geliştiriciler, bu nesneler üzerinde güncelleme yahut yeni veri ekleme vs. gibi işlemler gerçekleştireceksek bu takip mekanizmasına dokunmaksızın çalışmalarımıza devam etmeliyiz. Yok eğer elde ettiğimiz verileri sade ve sadece elde etmekle yetinecek ve bu veriler üzerinde herhangi bir güncelleme, ekleme yahut silme işlemi gerçekleştirmeyeceksek bu takip mekanizmasını kırmalıyız!

**Soru:**Entity framework failed concurrency crud işleminde nasıl tepki verir?

Defaut olarak concurent job'a karışan tüm transactionlar rollback alır.

**Soru:**Singleton, transient, scope nedir nasıl çalışır?

Yaşam süreleri farklıdır. Doğru şekilde kullanmanın hem mantıksal hem de performansa yönelik etkileri vardır.

Singleton ile nesne bir kez oluşur ve uygulama boyunca kullanılır.

Transient her çağrıda nesne tekrar oluşturulur ve iş bitiminde dispose olur.

Scoped request başından bitimine kadar hayatta kalır.

**Soru:**Exceptionları yakaladığınızda throw ile atmak veya throw new Exception() ile atmak arasında ne gibi farklar vardır?

throw ile yakalanan exception orijinal hali korunur ve yukarıya bu bilgiyi koruyarak gönderir. Diğer yöntem de orijinal hatayı yakalamakla birlikte stack trace bilgisini sıfırlar. throw daha mantıklı gözükürken bazı durumlarda kullanıcıya custom exception'lar da vermek isteyebiliriz. O yüzden new Exception() ile kullanmak durumunda kalırız (bunun yeni bir nesne oluşturduğunu ve nesne oluşturmanın yazılımdaki en büyük maliyetlerden olduğunu unutmayın.) [[Dahası](https://stackoverflow.com/questions/2999298/difference-between-throw-and-throw-new-exception)]

**Soru**: Mocklama nedir, neden yapılır?

Unit testler yaparken uygulamamızın tümünü ayakta tutmak/tutabilmek zor olacaktır. Bunun yerine uygulamamızın test ortamında çalışabileceği kadar fonksiyonları içeren sınıflar denir.

Selim Şenyurt ise şöyle bir tanım yapmıştır:” Ulaşamayacağımız servis veya veritabanlarını sanal olarak oluşturulan nesnelerdir.”

**Soru**: unit ve integration test nedir? Farklarından bahseder misiniz?

Unit test adından anlaşılacağı üzere sadece bir metodu test eden test şeklidir. Hızlıdır, mock kullanılır, servis, db gibi istekler yapılmaz. Diğer metodlardan gelen cevaplardan etkilenmeden kodlanması gerekir, bu yüzden kırılması nispeten daha zordur. Çalışma süreleri ms cinsinden olmalıdır. İntegration test ise birçok metodun birlikte çalışmasını test eder. Bir bakıma uçtan uça bir istek ve sonuçtan oluşur. Service, db isteklerini gerçekten de yapar, mocklama kullanılmaz. Çalışma süreleri ise unit testlere göre daha uzundur. [[Dahası](https://medium.com/software-development-turkey/birim-unit-test-ile-veri-k%C3%BCmeleri-xunit-inline-member-class-data-601b3fb4e723)] [[Dahası](https://medium.com/software-development-turkey/integration-test-net-core-xunit-web-application-factory-600ca6a52223)]

**Soru**: void methodları test etmenin zorluğundan bahseder misiniz?

Bir metodun sağlıklı olarak test edilebilmesi için bir sonuç döndürmesi gerekmektedir. Çünkü metod sonucunda bir değer döner. Gelen bu değeri, beklediğimiz değer ile karşılaştırırız ve bunlara göre bazı sonuçlar, hatalar veya mesajlar döndürürüz. void olan metodlar ise herhangi bir dönüş sağlamadığından bunları test etmek nispeten zor olsa da bunun için farklı yollar izlenebilir. [[Dahası](https://medium.com/software-development-turkey/void-metodlar%C4%B1-test-etmek-15b7b43a4c2c)]

**Soru**: static kod analiz araçları nedir, ne işe yarar, kullandığınız oldu mu?

Öncelikle kod analizi 2'ye ayrılır. Bunlar dinamik ve statik kod analizidir. Dinamik kod analizi run time’de yapılan analizdir. Statik ise kodu derlemeden yapılan analizdir. Bu sayede geliştirdiğiniz projede temiz kod yazımını kolaylaştırır, teknik borcunu azaltmak gibi birçok faydası bulunmaktadır.

SonarQube, Cast, Fortify gibi araçlar mevcuttur. Bunun yanında client bazlı çalışan vs extention olan sonarlint da mevcut. [[Dahası](https://medium.com/software-development-turkey/sonarqube-ile-kod-analizi-1acf2a3add55)] [[Dahası](https://medium.com/software-development-turkey/sonarlint-ile-refactoring-teknik-borcu-azaltma-7d3da6c92f19)]

Soru: 100, 200, 300, 400, 500 http kodları hakkında genel bilgi verebilir misin?

100 ile başlayanlar bilgilendirme, 200 ile başlayanlar başarılı olanlar, 300 ile başlayan yönlendirme ile ilgili olanlar, 400 ile başlayan client bazlı olanlar ve 500 ile başlayanlar ise server bazlı sonuçlar dönmektedir. [[Dahası](https://httpstatuses.com/)]

**Soru**: Dependecy injection hangi şekillerde yapılabilir?

1. Constructor Injection

2. Middleware Injection

3. Action Injection

4. View Injection

5. attribute injection

<https://tr.bitdegree.org/tutorial/html-mulakat-sorulari/>

<https://akillitavsan.com/yazilim-gelistirici-mulakat-sorulari/>

<https://hakanyalitekin.wordpress.com/2019/06/24/yazilim-mulakat-sorulari/>