**1)C# class can inherit multiple \_\_\_\_\_\_\_\_**

A: Classes

**B: Interfaces**

C: Abstract Classes

D: Static Classes

C # çoklu sınıf mirasını desteklemez. Bu sorunun üstesinden gelmek için (çoklu sınıf mirası elde etmek için) interface kullanır.

**Interfaceler** içlerine kod yazılmayan, yapıcı metod ekleyemeyeceğiniz, sadece tanımlamaların yapıldığı ve bu tanımlamalar doğrultusunda kendisinin rehberlik ettiği classların içerisinde yapıların oluşturulmasını (implement edilmesini) sağlayan bir yapıdır. Classlardan en büyük farkı ise bir class’a birçok interface ile rehberlik edebiliriz ama bir class ancak bir tane classtan türeyebilir. Implement edilen class tarafından bütün üyeler erişilebilir, içerisinde yalnızca metodlar ve propertyler tanımlanır, field tanımlanamaz. Sadece get ve set metodu içeren propertyler üzerinden değişken tanımlaması yapabilirsiniz.

**Abstract Class, ortak özellikli Class’lara Base(taban) Class olma görevini üstlenir.** Bu nedenle Abstract Class’dan **nesne türetilemez.** Bir sınıf sadece bir abstract sınıfı inherit edebilir ama bir sınıf birden fazla interface’i inherit edebilir bu yüzden C# da interfaceler kullanılır. Ayrıca abstract classlar çoklu kalıtımı desteklemezler. Abstract method, sadece Abstract Class’ların içerisinde tanımlanır ve Abstract Class’ı kalıtan sınıf tarafından override edilmek zorundadır. Oluşturduğumuz Abstract Class içerisine, Abstract Method yazılırken gövdesi yazılmaz ve Abstract Class’ımızı kalıtacağımız sınıfta Abstract Method’u override ederiz. Abstract class’lar erişim belirleyici olarak private, protected, private internal ile tanımlanamazlar.

**2)Which of the followings are value types in C#?**

A: Int32

B: Double

C: Decimal

**D: All of the above**

Bir veri türü, kendi bellek alanında bir veri değerini tutuyorsa bir Value Type’dır. Veri türlerinin değişkenlerinin doğrudan değerler içerdiği anlamına gelir. Aynı şekilde bool, byte, char, decimal, int, double, long bunlar da Value Type’dır.

**3)Which of the following is a reference type in C#?**

**A: String**

B: Long

C: Boolean

D: None of the above

Referans tipleri verinin kendisini tutmaz onun yerine verinin bulunduğu bellek adreslerini saklarlar. Array,Class ve Delegate de Reference Type’dır.

**4)What is Nullable type?**

A: It allows assignment of null to reference type.

**B: It allows assignment of null to value type.**

C: It allows assignment of null to static class.

D: None of the above.

Nullable Type bir değişkene boş değer atamamızı sağlar. Value Type değişkeni, bir bellek adresini değil, değeri tuttuğu için boş olamaz. Nullable Types yalnızca Value Type kullanabilir.

Syntax :

Nullable <data\_type> variable\_name = null;

Or

Data\_type? variable\_name = null;

Nullable Type değerine doğrudan erişilmez GetValueOrDefault() yöntemini kullanmalıyız, bu boş ise varsayılan değer alınır yanı sıfır.

HasValue ise nesneye bir değer atandıysa true, nesne null olarak atanmışsa false döndürür.

Nullable Type , tanımsız değeri de temsil edebilir.

**5)Struct is a \_\_\_\_\_.**

A: Reference Type

**B: Value Type**

C: Class Type

D: String Type

Struct C # da veri yapılarını temsil eden ValueType’dır. Struct deyimi, programınız için birden fazla üye içeren yeni bir veri türünü tanımlar. Bir struct nesnesi new operatörü kullanılarak oluşturulabilir. Bir yapının bireysel üyelerine nokta (.) Operatörü kullanılarak erişilebilir. Sınıflardan farklı olarak struct nesnesinin new operatörü kullanılmadan da oluşturulabilir.

MyStruct ms;

Ancak bu durumda, yapının tüm alanları atanmamış olarak kalır ve tüm alanlar başlatılana kadar nesne kullanılamaz.

**6)10 > 9 ? "10 is greater than 9” : "9 is greater than 10” is an example of \_\_\_\_\_\_\_**

**A: Ternary operator**

B: Conditional operator

C: Greater than operator

D: Inverse operatör

**Ternary operatör:** Basitleştirilmiş bir if-else ifadesi biçimidir. Üçlü operatörün ilk işlenen bir koşuldur, bir boole değeri döndürmelidir. İkinci ve üçüncü işlenenler, bir koşul temelinde yürütülen ifadelerdir. İlk işlenenin koşulu karşılanırsa ve true döndürürse, ikinci işlenen ifadesi çalıştırılır. Aksi takdirde, üçüncü işlenen ifadesi çağrılır.

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

**A: Int**

B: String

C: Boolean

D: All of the above

Enum: Bir numaralandırma türüdür. İnteger değerlerine sabit adlar atamak için kullanılır.

Değerler enum üyelere atanmamışsa, derleyici varsayılan olarak sıfırdan başlayarak her üyeye tamsayı değerleri atar. Enum üyerlinden birine değer atadığında diğer üyeler de sırayla otamatik olarak artırımlı değerler atar. Numaralandırma byte,short,int long herhangi bir veri türünde olabilir.

**8)What is indexer?**

**A: It allows an instance of a class to be indexed like an array**

B:It allows enumerator with class

C:It creates index for instances of a class.

D:None of the above.

Indexer: Bir sınıf veya Structın dizilere benzer şekilde indexlenmesini sağlar. Bir tam sayı değerine göre diziye alınması gerekmez. İndexerlar overloaded edebilirler, this indexer tanımlamak için kullanır. Değerleri atamak ve almak için basit get ve set erişimci yöntemlerine sahip genel bir sınıfı tanımlar

**9)String data type is \_\_\_\_\_\_.**

A: Mutable

**B: Immutable**

C: Static

D: Value type

String : Bir reference Type’dır, değişmezdir, null değerler içerebilir.

**10)An array in C# starts with \_\_\_\_\_ index.**

A: One

**B: Zero**

C: -1

D: None of the above

Array: Birden fazla veriyi aynı tür bir veri tipinde saklayabilmemize olanak sağlayan belirli bir uzunluğu olan veri yapısıdır. Dizi öğeleri, bellekte bitişik olarak depolanır. 0 dan başlar.Bir diziyi başlatma:

double[] ornekDizi = new double[10];

**11)Which of the following is right way of declaring an array?**

A: int[] intArray = new int[];

B: int intArray[] = new int[5];

**C: int[] intArray = new int[5];**

D: int[5] intArray = new int[];

Diziye değerler atayabilirsiniz:

double[] ornekDizi = { 2340.0, 4523.69, 3421.0};

Dizi oluşturabilir ve başlatabilirsiniz:

int [] ornekDizi = new int[5] { 99, 98, 92, 97, 95};

Dizinin boyutunu da atlayabilirsiniz

int [] ornekDizi = new int[] { 99, 98, 92, 97, 95};

Bir dizi değişkenini başka bir hedef dizi değişkenine kopyalayabilirsiniz. Böyle bir durum da, hem hedef hem de kaynak aynı hafıza konumuna işaret eder

int []ornekDizi = new int[] { 99, 98, 92, 97, 95};

int[] DiziX= ornekDizi;

**12)Which of the following is true for ReadOnly variables?**

**A: Value will be assigned at runtime.**

B: Value will be assigned at compile time.

C: Value will be assigned when it accessed first time

D: None of the above

**ReadOnly:**Metodun içinde değil yalnızca sınıf düzeyinde bildirilebilir. Sabit değeri reference variable ile ayarlanabilir. Runtime sabit değişkendir. Const ile farkı sadece tanımlandığı yerde değil, ayrıca constructor içinde de tanımlanabilir. (injection işlemlerinde çok sık kullanılır). Constructor dışında ise tanımlanamazlar.

**const**: Bir değişken tanımlaması sağlanır, fakat tanımlanan bu değişken daha sonra hiç bir şekilde değiştirilemez. Kesinlikle değişmeyeceğini bildiğiniz yerlerde kullanın. const ifadeler sadece primative(int, double,short) değerler ile birlikte kullanılır. Referans bazlı tanımlarla birlikte kullanılamazlar. Compile time sabit değişkenidir

**Static:** Static değişkenlere veya metodlara erişirken obje oluşturmamıza gerek yoktur. Static tanımlanan öğeler her defasında oluşturmazlar, bir kere oluştururlar ve program sonlanıncaya kadar yaşam döngüleri devam eder.

Const’u asla değişmeyecek bir değişken tanımında, Readonly ‘i değişeceğinden emin olamadığınız ama başka yerde değişmesini istemediğinizde, Static ‘i ise direk erişmek istiyorsanız kullanın.

**13)Which of the following statement is true?**

A: try block must be followed by catch or finally block or both.

B: finally block cannot include return statement.

C: try block can include another try block.

**D: All of the above.**

**14)Which of the following statement is true?**

A: A finally block may not execute every time

**B: A return or break statement cannot exit a finally block.**

C: A finally block can come before catch block.

D: Multiple finally block is possible.

**15)Func and Action are the types of \_\_\_\_\_\_.**

A: Utility function

**B: Delegate**

C: Event

D: Generic class

Delegate: Metodları işaret eden ve işaret ettikleri metodlarla aynı imzaya sahip olan yani aynı tipte parametrelere ve dönüş değerlerine sahip olan referans tipli yapılardır.

Bir temsilciyi, bir metodu işaret etmesi için kullanmak istediğimizde ise, çalışma zamanında onu new yapılandırıcısı ile oluşturur ve işaret etmesini istediğimiz metodu ona parametre olarak veririz .Örnek delegate:

public delegate void DenemeHandler(int a,int b);

Handler delegatelerde kullanılanılan yazılımcı geneleğidir yazılması zorunlu değildir. Delegeleri event tabanlı programlamada, asenkron programlamada, thread programlamada vb birçok yerde kullanırız.

**Action Delegate:** T tipinde yani herhangi bir tipte parametre alan ve void dönüşü yapan metodları işaret ediyor.

public delegate void Action<in T>(Tobj)

**Func Delegate** : Action’a benzer bir yapıdadır ama Actiondan farklı olarak bir dönüş değeri vardır. Func delegesinin tanımlarına baktığımızda iki farklı kullanım görebiliriz. Hiç parametre almadan bir değer dönebilir.

public delegate TResult Func<out TResult>()

Ya da bir parametre alıp bir değer dönebilir.

Public delegate TResult Func<in T,out TResult>(T arg)

**Predicate Delegate**: Tam olarak Func<T, bool> parametresinin yaptığı işi yapar. Yani herhangi bir tipteki değişkeni alır ve geriye boolean bir değer döner.

**16)Return type of Predicate <T>() is always a \_\_\_\_\_\_.**

A: Integer

B: String

**C: Boolean**

D: Void

**17)A partial class allows \_\_\_\_\_\_\_\_**

**A: Implementation of single class in multiple .cs files.**

B: Declaration of multiple classes in a single .cs file.

C: Implementation of multiple interfaces to single class.

D: Multiple class inheritance.

**Partial class** bize bir class' ı birden fazla class olarak bölmemize, constructor, değişken, property, metodları vs. düzenli bir şekilde ayrı ayrı oluşturmamızı sağlamaktadır.

Partial anahtar kelimesi sınıf, değişken tanımı, metot ve property oluşturulurken, kullanılmalıdır.

Partial class ile fiziksel olarak parça class'ların birleşmesi için class isimlerinin aynı olması gerekmektedir. Metodlar void döndürülmelidir. Birden fazla parça ile oluşan partial class' lar, çalışma zamanında tek bir class olarak bütün elemanları içerisinde barındırmaktadır

Büyük projeler üzerinde çalışırken, bir sınıfı ayrı dosyalar üzerine yaymak, birden çok programcının aynı anda üzerinde çalışmasını sağlar.

**18)LINQ stands for \_\_\_\_\_\_\_\_.**

**A: Language Integrated Query**

B: Local Integration Query

C: Language Included Query

D: Lazy Integrated Query

**LINQ** kodlarımız içerisinde, farklı dil kullanmadan, çeşitli ortamlardaki verileri sorgulayıp, filtrelemek için kullanılır.

Bir LINQ sorgusunun sonucunu çalıştırana kadar almayacaksınız

Derleme zamanı sözdizimi denetimi

**LINQ'nun Avantajları**

Farklı veri kaynaklarında ki veriler için ortak kullanım

Daha az kod

Okunabilir Kod

Farklı veri kaynaklarında ki ortak sorgulama standartı

Derleme esnasında nesne verilerinin kontrolünün sağlanabilmesi

**19)Data type of a variable declared using var will be assigned at \_\_\_\_\_\_\_.**

A: Runtime

**B: Compile time**

C: CLR time

D: Application Initialization time.

Var anahtarı, değişken tanımı yaparken tür belirtmeksizin tanım yapmamızı sağlamaktadır. Var ile tanımlanan bir değişkene atanan ilk değer program derlendiği anda değişkenin veri türünü belirlemektedir. var değişken tipi,diller arası,databaseler arası entegrasyonu sağlarken veri tipleri uyuşmazlığını gidermek için oluşturulmuş bir tiptir.

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.

Ornek: var isim = “Melike”;

Yukarıda var ile yapılan işlem aynı şekilde object ile de yapılabilmektedir.

Örnek: object isim= “Melike”;

 2 yapı arasındaki fark Object ile tanımlanan bir değişkene farklı türlerde değer atanabilmektedir.

Örnek;

Object isim;

isim= “Melike”;

isim=10;

isim=true;

Örnekte görüldüğü gibi aynı değişkene sırasıyla string, int ve bool değer türlerinde veriler atanmıştır. Ve derleme işlemi düzgün bir şekilde yapılacaktır. Fakat bu durum var ile tanımlanan bir değişken için geçerli değildir. Çünkü var ile tanımlı bir değişkene atanan ilk değer o değişkenin veri türünü oluşturacaktır ve sonrasında farklı veri türünde bir değer atanmak istenildiğinde derleme hatası verecektir.

**20)Which of the following is true for dynamic type in C#?**

A: It allows multiple time declaration of a variable.

B: It allows compile time type checking

**C: It escapes compile time type checking**

D: None of the above.

Derleme zamanı tür denetiminden kaçınmak için kullanılır. Derleyici dinamik tür değişkeninin türünü derleme zamanında kontrol etmez, bunun yerine derleyici türü çalışma zamanında alır. Dinamik tür değişkeni, dinamik anahtar sözcük kullanılarak oluşturulur.

dynamic value=158;

GetType () yöntemini kullanarak çalışma zamanında dinamik değişkenin gerçek türünü alabilirsiniz.

Dinamik tür için tür denetimi yoksa derleyici, derleme zamanında dinamik türe bir hata atmaz.

10.02.2021 Melike Sertkaya