

C#.Net Interview Questions and Answers

What Is NUnit?

Answer :: NUnit is a unit-testing framework for all .Net languages. Initially ported from JUnit, the current production release, version 2.2, is the fourth major release of this xUnit based unit testing tool for Microsoft .NET. It is written entirely in C# and has been completely redesigned to take advantage of many .NET language features, for example custom attributes and other reflection related capabilities. NUnit brings xUnit to all .NET languages. [Read More](#)

In the NUnit test framework, which attribute must adorn a test class in order for it to be picked up by the NUnit GUI?

Answer :: I think it will be TestFixtureAttribute

The test fixture attribute is applied to a class that contains a suite of related test cases. If an error occurs while initializing the fixture or if at least one of the test cases within the fixture fails, then the fixture itself will be deemed to have failed. Otherwise the fixture will pass. Output from the fixture, such as text written to the console, is captured by the framework and will be included in the test report.

A test fixture has no timeout by default. This may be changed using the TimeoutAttribute.

This attribute may be omitted whenever a test fixture class contains at least one test method or test parameter or when other MbUnit attributes are applied to the test fixture class. This is almost always the case unless for some reason you have an empty fixture.

The class must have a public default constructor. The class may not be static. [Read More](#)

When should you call the garbage collector in .NET?

Answer :: when Memory is low.... [Read More](#)

What is a satellite assembly?

Answer ::
When you write a multilingual or multi-cultural application in .NET, and want to distribute the core application separately from the localized modules, the localized assemblies that modify the core application are called satellite assemblies. [Read More](#)

What's a multicast delegate?

Answer ::
It's a delegate that points to and eventually fires off several methods. [Read More](#)

What's a delegate?

Answer :: A delegate object encapsulates a reference to a method. [Read More](#)

[What are the different ways a method can be overloaded?](#)

Answer :: Different data types, different number, and different order of parameter. [Read More](#)

[How is method overriding different from method overloading?](#)

No Accepted answer found [View Answer](#)

[What's the difference between an interface and abstract class?](#)

No Accepted answer found [View Answer](#)

[What is the difference between a Struct and a Class?](#)

No Accepted answer found [View Answer](#)

[What is an interface class?](#)

No Accepted answer found [View Answer](#)

[What's an abstract class?](#)

Answer :: Suppose you are implementing a new class that you want others to derive from. lets say...Car, the abstract class might look something like this:

```
public abstract class Car
{
    public abstract void RemoveAllFuses();

    public abstract void StartEngine();
}
```

as you see, abstract class has methods that are NOT implemented, but just declared, the deriving class should implement them.

The Car example here states an abstract Car, and it's exactly what it is, abstract, a car with no name and no functionality yet, but with some declaration on what it does (of course regular car has more than StartEngine and RemoveAllFuses).

now each car who derives from the abstract one must implement both methods.

for example:

```
class Seat : Car
{
```

```
public override void RemoveAllFuses()
```

```
{
```

```
}
```

```
public override void StartEngine()
```

```
{
```

```
}
```

```
}
```

thats great...but now the question:

What should you do in order NOT to let a certain method withing the derived to be called?

eg, what whould you do to prevent this:

```
class MyCar : Seat
```

```
{
```

```
static void Main(string[] args)
```

```
{
```

```
MyCar c= new MyCar();
```

```
try
```

```
{
```

```
c.RemoveAllFuses();
```

```
}
```

```
catch (Exception e)
{
    System.Console.WriteLine(e.Message);
}
}
```

you ask why? well, i say, Seat Car maker did not want to implement RemoveAllFuses.

the answer is really in the code, in the try block.

in the Seat Class, we should throw an exception in order to make that method unavailable, and at the same time, we have actually implemented it. this is what you do if you DONT NEED ALL METHODS from a derived class.

the modified Seat class is:

```
class Seat : Car
{
    public override void RemoveAllFuses()
    {
        throw new MethodAccessException();
    }
    public override void StartEngine()
    {
    }
}
```

MethodAccessException is being explained as "Attempt to access the method failed"...as simple as that.

.... [Read More](#)

[What's the C# syntax to catch any possible exception?](#)

No Accepted answer found [View Answer](#)