**1) What is a class ?**  
  
A class is the generic definition of what an object is. A Class describes all the attributes of the object, as well as the methods that implement the behavior of the member object. In other words, class is a template of an object. For ease of understanding a class, we will look at an example. In the class Employee given below, Name and Salary are the attributes of the class Person. The Setter and Getter methods are used to store and fetch data from the variable.

public class Employee

{

private String name;

private String Salary;

public String getName()

{

return name;

}

public void setName(String name)

{

this.name = name;   
}

public String getSalary ()

{

return Salary;

}

public void setSalary (String Salary)

{

this. Salary = Salary;

}

}

**2) What is an Object?**

An object is an instance of a class. It contains real values instead of variables. For example, let us create an instance of the class Employee called “John”.

Employee John= new Employee();

Now we can access all the methods in the class “Employee” via object “John” as shown below.

John.setName(“XYZ”);

**3) What are the Access Modifiers in C# ?**

Different Access Modifier are - Public, Private, Protected, Internal, Protected Internal

* Public – When a method or attribute is defined as Public, it can be accessed from any code in the project. For example, in the above Class “Employee” getName() and setName() are public.
* Private - When a method or attribute is defined as Private, It can be accessed by any code within the containing class only. For example, in the above Class “Employee” attributes name and salary can be accessed within the Class Employee Only. If an attribute or class is defined without access modifiers, it's default access modifier will be private.
* Protected - When attribute and methods are defined as protected, it can be accessed by any method in the inherited classes and any method within the same class. The protected access modifier cannot be applied to classes and interfaces. Methods and fields in a interface can't be declared protected.
* Internal – If an attribute or method is defined as Internal, access is restricted to classes within the current project assembly.
* Protected Internal – If an attribute or method is defined as Protected Internal, access is restricted to classes within the current project assembly and types derived from the containing class.

**Must Read** - [.Net Framework Interview Questions and Answers](http://a4academics.com/interview-questions/52-dot-net-interview-questions/421-net-interview-questions-and-answers-on-net-framework)

**4) Explain Static Members in C# ?**

If an attribute's value had to be same across all the instances of the same class, the static keyword is used. For example, if the Minimum salary should be set for all employees in the employee class, use the following code.

private static double MinSalary = 30000;

To access a private or public attribute or method in a class, at first an object of the class should be created. Then by using the object instance of that class, attributes or methods can be accessed. To access a static variable, we don't want to create an instance of the class containing the static variable. We can directly refer that static variable as shown below.

double var = Employee.MinSalary ;

**Must Read** - [ADO.Net Interview Questions and Answers](http://a4academics.com/interview-questions/52-dot-net-interview-questions/488-ado-net-interview-questions-and-answers) 

**5) What is Reference Type in C# ?**

Let us explain this with the help of an example. In the code given below,

Employee emp1;

Employee emp2 = new Employee();

emp1 = emp2;

Here emp2 has an object instance of Employee Class. But emp1 object is set as emp2. What this means is that the object emp2 is referred in emp1, rather than copying emp2 instance into emp1. When a change is made in emp2 object, corresponding changes can be seen in emp1 object.

**6) Define Property in C# ?**

Properties are a type of class member, that are exposed to the outside world as a pair of Methods. For example, for the static field Minsalary, we will Create a property as shown below.

private double minimumSalary;

public static double MinSalary

{

get

{

return minimumSalary;

}

set

{

minimumSalary = value;

}

}

So when we execute the following lines code

double minSal = Employee.MinSalary;

get Method will get triggered and value in minimumSalary field will be returned. When we execute,

Employee. MinSalary = 3000;

set Method will get triggered and value will be stored in minimumSalary field.

**Don't Miss** - [Database Interview Questions and Answers](http://a4academics.com/interview-questions/53-database-and-sql/411-sql-interview-questions-and-answers-database)

**7) Explain Overloading in C# ?**

When methods are created with the same name, but with different signature its called overloading. For example, WriteLine method in console class is an example of overloading. In the first instance, it takes one variable. In the second instance, “WriteLine” method takes two variable.

Console.WriteLine(x);

Console.WriteLine("The message is {0}", Message);

Different types of overloading in C# are

* Constructor overloading
* Function overloading
* Operator overloading

**Must Read** - [SQL Query Interview Questions and Answers](http://a4academics.com/interview-questions/53-database-and-sql/397-top-100-database-sql-interview-questions-and-answers-examples-queries)

**8) What is Constructor Overloading in C# .net ?**

In Constructor overloading, n number of constructors can be created for the same class. But the signatures of each constructor should vary. For example

public class Employee

{

public Employee()

{ }

public Employee(String Name)

{ }

}

**Also Read** - [ASP.Net Interview Questions and Answers](http://a4academics.com/interview-questions/52-dot-net-interview-questions/706-asp-net)

**9) What is Function Overloading in C# .net ?**

In Function overloading, n number of functions can be created for the same class. But the signatures of each function should vary. For example

public class Employee

{

public void Employee()

{ }

public void Employee(String Name)

{ }

}

**Also Read** - [ASP.Net MVC Interview Questions and Answers](http://a4academics.com/interview-questions/52-dot-net-interview-questions/713-asp-net-mvc)

**10) What is Operator Overloading in C# .net ?**

We had seen function overloading in the previous example. For operator Overloading, we will have a look at the example given below. We had defined a class rectangle with two operator overloading methods.

class Rectangle

{

private int Height;

private int Width;

public Rectangle(int w,int h)

{

Width=w;

Height=h;

}

public static bool operator >(Rectangle a,Rectangle b)

{

return a.Height > b.Height ;

}

public static bool operator <(Rectangle a,Rectangle b)

{

return a.Height < b.Height ;

}

}

Let us call the operator overloaded functions from the method given below. When first if condition is triggered, the first overloaded function in the rectangle class will be triggered. When second if condition is triggered, the second overloaded function in the rectangle class will be triggered.

public static void Main()

{

Rectangle obj1 =new Rectangle();

Rectangle obj2 =new Rectangle();

if(obj1 > obj2)

{

Console.WriteLine("Rectangle1 is greater than Rectangle2");

}

if(obj1 < obj2)

{

Console.WriteLine("Rectangle1 is less than Rectangle2");

}

}

**11) What is Data Encapsulation ?**

Data Encapsulation is defined as the process of hiding the important fields from the end user. In the above example, we had used getters and setters to set value for MinSalary. The idea behind this is that, private field “minimumSalary” is an important part of our classes. So if we give a third party code to have complete control over the field without any validation, it can adversely affect the functionality. This is inline with the OOPS Concept that an external user should know about the what an object does. How it does it, should be decided by the program. So if a user set a negative value for MinSalary, we can put a validation in the set method to avoid negative values as shown below

set

{

if(value > 0)

{

minSalary = value;

}

}

**`  
12) Explain Inheritance in C# ?**

In object-oriented programming (OOP), inheritance is a way to reuse code of existing objects. In inheritance, there will be two classes - base class and derived classes. A class can inherit attributes and methods from existing class called base class or parent class. The class which inherits from a base class is called derived classes or child class. For more clarity on this topic, let us have a look at 2 classes shown below. Here Class Car is Base Class and Class Ford is derived class.

class Car

{

public Car()

{

Console.WriteLine("Base Class Car");

}

public void DriveType()

{

Console.WriteLine("Right Hand Drive");

}

}

class Ford : Car

{

public Ford()

{

Console.WriteLine("Derived Class Ford");

}

public void Price()

{

Console.WriteLine("Ford Price : 100K $");

}

}

When we execute following lines of code ,

Ford CarFord = new Ford();

CarFord.DriveType();

CarFord.Price();

Output Generated is as given below.

Base Class Car

Derived Class Ford

Right Hand Drive

Ford Price : 100K $

What this means is that, all the methods and attributes of Base Class car are available in Derived Class Ford. When an object of class Ford is created, constructors of the Base and Derived class get invoked. Even though there is no method called DriveType() in Class Ford, we are able to invoke the method because of inheriting Base Class methods to derived class.

**13) Can Multiple Inheritance implemented in C# ?**

In C#, derived classes can inherit from one base class only. If you want to inherit from multiple base classes, use interface.

**14) What is Polymorphism in C# ?**

The ability of a programming language to process objects in different ways depending on their data type or class is known as Polymorphism. There are two types of polymorphism

* Compile time polymorphism. Best example is Overloading
* Runtime polymorphism. Best example is Overriding

**15) Explain the use of Virtual Keyword in C# ?**

When we want to give permission to a derived class to override a method in base class, Virtual keyword is used. For example. lets us look at the classes Car and Ford as shown below.

class Car

{

public Car()

{

Console.WriteLine("Base Class Car");

}

public virtual void DriveType()

{

Console.WriteLine("Right Hand Drive");

}

}

class Ford : Car

{

public Ford()

{

Console.WriteLine("Derived Class Ford");

}

public void Price()

{

Console.WriteLine("Ford Price : 100K $");

}

public override void DriveType()

{

Console.WriteLine("Right Hand ");

}

}

When following lines of code get executed

Car CarFord = new Car();

CarFord.DriveType();

CarFord = new Ford();

CarFord.DriveType();

Output is as given below.

Base Class Car

Right Hand Drive

Base Class Car

Derived Class Ford

Right Hand

**16) What is overriding in c# ?**

To override a base class method which is defined as virtual, Override keyword is used. In the above example, method DriveType is overridden in the derived class.

**17) What is Method Hiding in C# ?**

If the derived class doesn't want to use methods in the base class, derived class can implement the same method in a derived class with same signature. For example, in the classes given below, DriveType() is implemented in the derived class with same signature. This is called Method Hiding.

class Car

{

public void DriveType()

{

Console.WriteLine("Right Hand Drive");

}

}

class Ford : Car

{

public void DriveType()

{

Console.WriteLine("Right Hand ");

}

}

**18) What is Abstract Class in C#?**

If we don't want a class to be instantiated, define the class as abstract. An abstract class can have abstract and non abstract classes. If a method is defined as abstract, it must be implemented in derived class. For example, in the classes given below, method DriveType is defined as abstract.

abstract class Car

{

public Car()

{

Console.WriteLine("Base Class Car");

}

public abstract void DriveType();

}

class Ford : Car

{

public void DriveType()

{

Console.WriteLine("Right Hand ");

}

}

Method DriveType get implemented in derived class.

**19) What is Sealed Classes in c# ?**

If a class is defined as Sealed, it cannot be inherited in derived class. Example of a sealed class is given below.

public sealed class Car

{

public Car()

{

Console.WriteLine("Base Class Car");

}

public void DriveType()

{

Console.WriteLine("Right Hand ");

}

}

**20) What is an Interface in C# ?**

An interface is similar to a class with method signatures. There wont be any implementation of the methods in an Interface. Classes which implement interface should have an implementation of methods defined in the abstract class.

**21) What is a Constructor in C# ?**

Constructor is a special method that get invoked/called automatically, whenever an object of a given class gets instantiated. In our class car, constructor is defined as shown below

public Car()

{

Console.WriteLine("Base Class Car");

}

When ever an instance of class car is created from the same class or its derived class(Except Few Scenarios), Constructor get called and sequence of code written in the constructor get executed.

interface Breaks

{

void BreakType();

}

interface Wheels

{

void WheelType();

}

class Ford : Breaks, Wheels

{

public Ford()

{

Console.WriteLine("Derived Class Ford");

}

public void Price()

{

Console.WriteLine("Ford Price : 100K $");

}

public void BreakType()

{

Console.WriteLine("Power Break");

}

public void WheelType()

{

Console.WriteLine("Bridgestone");

}

}

**22) What is a Destructor in C# ?**

Destructor is a special method that get invoked/called automatically whenever an object of a given class gets destroyed. Main idea behind using destructor is to free the memory used by the object.

**What are namespaces, and how they are used?**

Namespaces are used to organize classes within the .NET Framework. They dictate the logical structure of the code. They are analogous to Java packages, with the key difference being Java packages define the physical layout of source files (directory structure) while .NET namespaces do not. However, many developers follow this approach and organize their C# source files in directories that correlate with namespaces. The .NET Framework has namespaces defined for its many classes, such as System.Xml—these are utilized via the using statement. Namespaces are assigned to classes via the namespace keyword.

**What is a constructor?**

A constructor is a class member executed when an instance of the class is created. The constructor has the same name as the class, and it can be overloaded via different signatures. Constructors are used for initialization chores.

**What is the GAC, and where is it located?**

The GAC is the Global Assembly Cache. Shared assemblies reside in the GAC; this allows applications to share assemblies instead of having the assembly distributed with each application. Versioning allows multiple assembly versions to exist in the GAC—applications can specify version numbers in the config file. The gacutil command line tool is used to manage the GAC.

**Why are strings in C# immutable?**

Immutable means string values cannot be changed once they have been created. Any modification to a string value results in a completely new string instance, thus an inefficient use of memory and extraneous garbage collection. The mutable System.Text.StringBuilder class should be used when string values will change.

**What is DLL Hell, and how does .NET solve it?**

DLL Hell describes the difficulty in managing DLLs on a system; this includes multiple copies of a DLL, different versions, and so forth. When a DLL (or assembly) is loaded in .NET, it is loaded by name, version, and certificate. The assembly contains all of this information via its metadata. The GAC provides the solution, as you can have multiple versions of a DLL side-by-side.

**How are methods overloaded?**

Methods are overloaded via different signatures (number of parameters and types). Thus, you can overload a method by having different data types, different number of parameters, or a different order of parameters.

**How do you prevent a class from being inherited?**

The sealed keyword prohibits a class from being inherited.

**What is the execution entry point for a C# console application?**

The Main method.

**How do you initiate a string without escaping each backslash?**

Y++++++0ou put an @ sign in front of the double-quoted string.

String ex = @"This has a carriage return\r\n"

**What is the difference between a struct and a class?**

Structs cannot be inherited. Structs are passed by value and not by reference. Structs are stored on the stack not the heap. The result is better performance with Structs.

**What is a singleton?**

A singleton is a design pattern used when only one instance of an object is created and shared; that is, it only allows one instance of itself to be created. Any attempt to create another instance simply returns a reference to the first one. Singleton classes are created by defining all class constructors as private. In addition, a private static member is created as the same type of the class, along with a public static member that returns an instance of the class. Here is a basic example:

public class SingletonExample {

private static SingletonExample \_Instance;

private SingletonExample () { }

public static SingletonExample GetInstance() {

if (\_Instance == null) {

\_Instance = new SingletonExample ();

}

return \_Instance;

}

}

**What is boxing?**

[Boxing](http://www.techrepublic.com/article/put-up-your-fists-and-start-boxing-with-net/5766513) is the process of explicitly converting a value type into a corresponding reference type. Basically, this involves creating a new object on the heap and placing the value there. Reversing the process is just as easy with unboxing, which converts the value in an object reference on the heap into a corresponding value type on the stack. The unboxing process begins by verifying that the recipient value type is equivalent to the boxed type. If the operation is permitted, the value is copied to the stack.

**1.  What is object-oriented programming (OOP) Language?**  
Object-oriented programming (OOP) is a programming language model organized around objects rather than "actions" and data rather than logic. Historically, a program has been viewed as a logical procedure that takes input data, processes it, and produces output data.  
  
**2. Explain about C# Language.**  
C# is a OOPs language, .net framework use to compiled it, to generate machine code.  
  
**3. Types of comments in C#?**  
  
Single line comments  
// for single line comments  
  
Multiple line comments  
/\* for multi line comments \*/  
  
XML tags comments  
  
/// XML tags displayed in a code comment  
  
  
**4. Top reason to use C# language?**  
 Modern, general-purpose programming language  
Object oriented.  
Component oriented.  
Easy to learn.  
Structured language.  
It produces efficient programs.  
It can be compiled on a variety of computer platforms.  
Part of .Net Framework.  
  
**5. feature of C# language?**  
Boolean Conditions  
Automatic Garbage Collection  
Standard Library  
Assembly Versioning  
Properties and Events  
Delegates and Events Management  
Easy-to-use Generics  
Indexers  
Conditional Compilation  
Simple Multithreading  
LINQ and Lambda Expressions  
Integration with Windows

**Also Read:**[**Top 50 Common Job Interview Questions and answers**](http://www.itechaleart.com/2014/04/top-50-common-interview-qa.html)

**6. What is a Class?**  
a set or category of things having some property or attribute in common and differentiated from others by kind, type, or quality.  
  
**7. What is object?**  
Objects are created from Classes, in C#, is an instance of a class that is created dynamically. Object is also a keyword that is an alias for the predefined type System.  
  
**8. What is Constructors, explain with syntax**  
A is special method of the class that will be automatically invoked when an instance of the class is created is called as constructor.  
  
Constructors are mainly used to initialize private fields of the class while creating an instance for the class.  
  
When you are not creating a constructor in the class, then compiler will automatically create a default constructor in the class that initializes all numeric fields in the class to zero and all string and object fields to null.  
  
Syntax.  
[Access Modifier] ClassName([Parameters])  
{  
}  
 **9. Types of Constructors**  
Basically constructors are 5 types those are  
Default Constructor  
Parameterized Constructor  
Copy Constructor  
Static Constructor  
Private Constructor

**10. index value of the first element in an array?**  
first element is 0 (zero). In a Array.

**Also Read:**[**Android Interview Ques & Ans for Fresher**](http://www.itechaleart.com/2014/04/android-interview-ques-for-fresher.html)

**11. Different between method overriding and  method overloading?**  
In Overriding methods it will create two or more methods with same name and same parameter in different classes.  
  
while Overloading it will create more then one method with same name but different parameter in same class.  
 **12. Explain use of Abstract and Sealed Classes in C#?**  
The abstract keyword enables you to create classes and class members that are incomplete and must be implemented in a derived class.  
  
The sealed keyword enables you to prevent the inheritance of a class or certain class members that were previously marked virtual.  
  
**13. What is Static Classes?**  
A static class is basically the same as a non-static class, but there is one difference: a static class cannot be instantiated.  
In other words, you cannot use the new keyword to create a variable of the class type. Because there is no instance variable, you access the members of a static class by using the class name itself.  
  
**14. Explain Static Class Members.**  
A non-static class can contain static methods, fields, properties, or events.  
  
The static member is callable on a class even when no instance of the class has been created. The static member is always accessed by the class name, not the instance name. Only one copy of a static member exists, regardless of how many instances of the class are created.  
  
Static methods and properties cannot access non-static fields and events in their containing type, and they cannot access an instance variable of any object unless it is explicitly passed in a method parameter.  
  
**15. Which are Access Modifiers available in C#?**  
All types and type members have an accessibility level, which controls whether they can be used from other code in your assembly or other assemblies.  
  
You can use the following access modifiers to specify the accessibility of a type or member when you declare it:  
**public:** The type or member can be accessed by any other code in the same assembly or another assembly that references it.  
**private:** The type or member can be accessed only by code in the same class or struct.  
**protected:** The type or member can be accessed only by code in the same class or struct, or in a class that is derived from that class.  
**internal:** The type or member can be accessed by any code in the same assembly, but not from another assembly.

**Also Read:**[**Top 20 JSP Interview Questions and Answers**](http://www.itechaleart.com/2014/07/jsp-interview-qa.html)

**16. Data Types in C#?**  
bool, byte , char, decimal , double, float, int, long, sbyte , short, uint, ulong, ushort.  
  
More question coming soon.. we are updating our list of ques and answer... :)  
keep wait and watch for few days.

**1. What is C#?**

C# is an object oriented, type safe and managed language that is compiled by .Net framework to generate Microsoft Intermediate Language.

**2. What are the types of comment in C# with examples?**

Single line

Eg:



|  |  |
| --- | --- |
| 1 | //This is a Single line comment |

ii. Multiple line (/\* \*/)

Eg:



|  |  |
| --- | --- |
| 1  2  3 | /\*This is a multiple line comment  We are in line 2  Last line of comment\*/ |

iii. XML Comments (///).

Eg:



|  |  |
| --- | --- |
| 1  2  3 | /// summary;  ///  Set error message for multilingual language.  /// summary |

**3. Can multiple catch blocks be executed?**

No, Multiple catch blocks can’t be executed. Once the proper catch code executed, the control is transferred to the finally block and then the code that follows the finally block gets executed.

**4. What is the difference between public, static and void?**

Public declared variables or methods are accessible anywhere in the application. Static declared variables or methods are globally accessible without creating an instance of the class. The compiler stores the address of the method as the entry point and uses this information to begin execution before any objects are created. And Void is a type modifier that states that the method or variable does not return any value.

**5. What is an object?**

[](http://career.guru99.com/wp-content/uploads/2012/04/C-Sharp-Interview-Questions.jpg)

An object is an instance of a class through which we access the methods of that class. “New” keyword is used to create an object. A class that creates an object in memory will contain the information about the methods, variables and behavior of that class.

**6. Define Constructors?**

A constructor is a member function in a class that has the same name as its class. The constructor is automatically invoked whenever an object class is created. It constructs the values of data members while initializing the class.

**7. What is Jagged Arrays?**

The array which has elements of type array is called jagged array. The elements can be of different dimensions and sizes. We can also call jagged array as Array of arrays.

**8. What is the difference between ref & out parameters?**

An argument passed as ref must be initialized before passing to the method whereas out parameter needs not to be initialized before passing to a method.

**9. What is the use of using statement in C#?**

The using block is used to obtain a resource and use it and then automatically dispose of when the execution of block completed.

**10. What is serialization?**

When we want to transport an object through network then we have to convert the object into a stream of bytes. The process of converting an object into a stream of bytes is called Serialization. For an object to be serializable, it should inherit ISerialize Interface.  
De-serialization is the reverse process of creating an object from a stream of bytes.

**11. Can “this” be used within a static method?**

We can’t use ‘This’ in a static method because we can only use static variables/methods in a static method.

**12. What is difference between constants and read-only?**

Constant variables are declared and initialized at compile time. The value can’t be changed after wards. Read-only variables will be initialized only from the Static constructor of the class. Read only is used only when we want to assign the value at run time.

**13. What is an interface class?**

Interface is an abstract class which has only public abstract methods and the methods only have the declaration and not the definition. These abstract methods must be implemented in the inherited classes.

**14. What are value types and reference types?**

Value types are stored in the Stack whereas reference types stored on heap.  
Value types:



|  |  |
| --- | --- |
| 1 | int, enum , byte, decimal, double, float, long |

Reference Types:



|  |  |
| --- | --- |
| 1 | string , class, interface, object |

**15. What are Custom Control and User Control?**

Custom Controls are controls generated as compiled code (Dlls), those are easier to use and can be added to toolbox. Developers can drag and drop controls to their web forms. Attributes can be set at design time. We can easily add custom controls to Multiple Applications (If Shared Dlls), If they are private then we can copy to dll to bin directory of web application and then add reference and can use them.  
User Controls are very much similar to ASP include files, and are easy to create. User controls can’t be placed in the toolbox and dragged – dropped from it. They have their design and code behind. The file extension for user controls is ascx.

**16. What are sealed classes in C#?**

We create sealed classes when we want to restrict the class to be inherited. Sealed modifier used to prevent derivation from a class. If we forcefully specify a sealed class as base class then a compile-time error occurs.

**17. What is method overloading?**

Method overloading is creating multiple methods with the same name with unique signatures in the same class. When we compile, the compiler uses overload resolution to determine the specific method to be invoke.

**18. What is the difference between Array and Arraylist?**

In an array, we can have items of the same type only. The size of the array is fixed. An arraylist is similar to an array but it doesn’t have a fixed size.

**19. Can a private virtual method be overridden?**

No, because they are not accessible outside the class.

**20. Describe the accessibility modifier “protected internal”.**

Protected Internal variables/methods are accessible within the same assembly and also from the classes that are derived from this parent class.

**21. What are the differences between System.String and System.Text.StringBuilder classes?**

System.String is immutable. When we modify the value of a string variable then a new memory is allocated to the new value and the previous memory allocation released. System.StringBuilder was designed to have concept of a mutable string where a variety of operations can be performed without allocation separate memory location for the modified string.

**22. What’s the difference between the System.Array.CopyTo() and System.Array.Clone() ?**

Using Clone() method, we creates a new array object containing all the elements in the original array and using CopyTo() method, all the elements of existing array copies into another existing array. Both the methods perform a shallow copy.

**23. How can we sort the elements of the array in descending order?**

Using Sort() methods followed by Reverse() method.

**24. Write down the C# syntax to catch exception?**

To catch an exception, we use try catch blocks. Catch block can have parameter of system.Exception type.

Eg:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | Try  {  GetAllData();  }  catch(Exception ex)  {  } |

In the above example, we can omit the parameter from catch statement.

**25.   What’s the difference between an interface and abstract class?**

Interfaces have all the methods having only declaration but no definition. In an abstract class, we can have some concrete methods. In an interface class, all the methods are public. An abstract class may have private methods.

**26. What is the difference between Finalize() and Dispose() methods?**

Dispose() is called when we want for an object to release any unmanaged resources with them. On the other hand Finalize() is used for the same purpose but it doesn’t assure the garbage collection of an object.

**27. What are circular references?**

Circular reference is situation in which two or more resources are interdependent on each other causes the lock condition and make the resources unusable.

**28. What are generics in C#.NET?**

Generics are used to make reusable code classes to decrease the code redundancy, increase type safety and performance. Using generics, we can create collection classes. To create generic collection, System.Collections.Generic namespace should be used instead of classes such as ArrayList in the System.Collections namespace. Generics promotes the usage of parameterized types.

**29. What is an object pool in .NET?**

An object pool is a container having objects ready to be used. It tracks the object that is currently in use, total number of objects in the pool. This reduces the overhead of creating and re-creating objects.

**30. List down the commonly used types of exceptions in .Net?**

ArgumentException, ArgumentNullException , ArgumentOutOfRangeException, ArithmeticException, DivideByZeroException ,OverflowException , IndexOutOfRangeException ,InvalidCastException ,InvalidOperationException , IOEndOfStreamException , NullReferenceException , OutOfMemoryException , StackOverflowException etc.

**31. What are Custom Exceptions?**

Sometimes there are some errors that need to be handeled as per user requirements. Custom exceptions are used for them and are used defined exceptions.

**32. What are delegates?**

Delegates are same are function pointers in C++ but the only difference is that they are type safe unlike function pointers. Delegates are required because they can be used to write much more generic type safe functions.

**33. How do you inherit a class into other class in C#?**

Colon is used as inheritance operator in C#. Just place a colon and then the class name.



|  |  |
| --- | --- |
| 1 | public class DerivedClass : BaseClass |

**34. What is the base class in .net from which all the classes are derived from?**



|  |  |
| --- | --- |
| 1 | System.Object |

**35. What is the difference between method overriding and method overloading?**

In method overriding, we change the method definition in the derived class that changes the method behavior. Method overloading is creating a method with the same name within the same class having different signatures.

**36. What are the different ways a method can be overloaded?**

Methods can be overloaded using different data types for parameter, different order of parameters, and different number of parameters.

**37. Why can’t you specify the accessibility modifier for methods inside the interface?**

In an interface, we have virtual methods that do not have method definition. All the methods are there to be overridden in the derived class. That’s why they all are public.

**38. How can we set class to be inherited, but prevent the method from being over-ridden?**

Declare the class as public and make the method sealed to prevent it from being overridden.

**39. What happens if the inherited interfaces have conflicting method names?**

Implement is up to you as the method is inside your own class. There might be problem when the methods from different interfaces expect different data, but as far as compiler cares you’re okay.

**40. What is the difference between a Struct and a Class?**

Structs are value-type variables and classes are reference types. Structs stored on the stack, causes additional overhead but faster retrieval. Structs cannot be inherited.

**41. How to use nullable types in .Net?**

Value types can take either their normal values or a null value. Such types are called nullable types.



|  |  |
| --- | --- |
| 1  2  3  4 | Int? someID = null;  If(someID.HasVAlue)  {  } |

**42. How we can create an array with non-default values?**

We can create an array with non-default values using Enumerable.Repeat.

**43. What is difference between is and as operators in c#?**

“is” operator is used to check the compatibility of an object with a given type and it returns the result as Boolean.

“as” operator is used for casting of object to a type or a class.

**44. What’s a multicast delegate?**

A delegate having multiple handlers assigned to it is called multicast delegate. Each handler is assigned to a method.

**45. What are indexers in C# .NET?**

Indexers are known as smart arrays in C#. It allows the instances of a class to be indexed in the same way as array.

Eg:



|  |  |
| --- | --- |
| 1 | public int this[int index]    // Indexer declaration |

**46. What is difference between the “throw” and “throw ex” in .NET?**

“Throw” statement preserves original error stack whereas “throw ex” have the stack trace from their throw point. It is always advised to use “throw” because it provides more accurate error information.

**47. What are C# attributes and its significance?**

C# provides developers a way to define declarative tags on certain entities eg. Class, method etc. are called attributes. The attribute’s information can be retrieved at runtime using Reflection.

**48. How to implement singleton design pattern in C#?**

In singleton pattern, a class can only have one instance and provides access point to it globally.

Eg:



|  |  |
| --- | --- |
| 1  2  3  4 | Public sealed class Singleton  {  Private static readonly Singleton \_instance = new Singleton();  } |

**49. What is the difference between directcast and ctype?**

DirectCast is used to convert the type of an object that requires the run-time type to be the same as the specified type in DirectCast.

Ctype is used for conversion where the conversion is defined between the expression and the type.

**50. Is C# code is managed or unmanaged code?**

C# is managed code because Common language runtime can compile C# code to Intermediate language.