Scenario Based Questions in Microsoft Technologies - .Net with Answers

Hai Friends,

As we know that getting more experience and then going for the interview requires more knowledge and if we see, it requires more practical knowledge.

So in the continuation, I am trying to post few questions of OOPS which are scenario based and will be helpful when you are going for the interviews with 3+ years experience. These questions will definitely make you to think something more which you were thinking before about the answer of a particular question. With the hope that these questions and answers will be helpful to you, I am posting them and will try to update and include more in the future.

These questions are scenario based questions in .Net technologies which will help to prepare for the interviews. Few questions are related to OOP's concepts, and then few on Garbage Collector and memory related. So you can prepare them accordingly. These questions will be useful for those who are having the 3+ years experience and looking out for the opportunities in good companies.

1. How the Encapsulation is different from Abstraction? I think both are used to hide the unnecessary details? Then how they are different?

Ans.

Yes, Both Encapsulation and Abstraction do the same thing but with few differences.

Encapsulation mainly encapsulates the object and so hides the details as well as it binds the data.

So Encapsulation = Hiding + Binding the data

How it hides the data? Real-time Example?

Take the example of n-Tier application where we have an additional layer called Business Objects. This layer

contains all the entities with their properties. Take an entity name: Employee. This Employee will have the

class name "EmployeeBO.cs" and contains the public properties like EmpId, EmpName, Sal ets

EmployeeBO.cs

```
public class EmployeeBO
{
  private int _empId;
  public int EmpId
  {
    get { return _empId; }
    set { _empId = value; }
  }
}
```

So this is the presentation of one property in the Business object class. Now wherever we want this attribute,

We just need to create the object of this class and set/get the value of EmpId as:

```
// set the EmpId
EmployeeBo objEmployeeBO = new EmployeeBO();
objEmployeeBO.EmpId = 101;

// get the EmpId
int empId = objEmployeeBO.EmpID;
```

Now the question is where its setting or getting the value?

EmpId is the public property in the EmployeeBO class and contain no value. Only _empId contains the value

which is private and so it is not accessible.
So binding of the data happens to the EmpId through _empID and hiding happends through _empId which is
private. The Data is accessed through the Public property while the actual data is in private variable. So
binding + hiding using Encapsulation.

Abstraction is to ignore the unnecessary details and get the required details. So it also hides the unnecessary details. How?

Abstract class is the special type of class which can contain the abstract and concrete members. If we define
The member (Method/Property) as abstract, it must be overrides to the child class. We are not bothering here About the non-abstract or concrete members. Which is an unnecessary detail? If we ad an additional concrete Member in the abstract class, we need not to do anything in the child class. But if we add the abstract members, we must have to override it. So abstract doesn't care about the concrete members which are unnecessary for it and so it hides those details.

2. What do you mean by early binding and late binding in the object bindings? Which is good? Please give me a scenario where you have used the early binding and late binding concepts in your application?

Ans. In .Net, the early binding and last binding concepts comes under the polymorphism. As we know that there are 2 types of polymorphism-

1. Compile Time polymorphism
2. Run time polymorphism

The **Compile Time polymorphism** also called as the Overloading where we have the same method name with different behaviors. By implementing the multiple prototype of the same method, we can achieve the behavior of the Overloading.

Also this behavior is valid only for a single class. It means we should have all the overloaded methods in the same class.

e.g.

```
/// <summary>
/// Early Binding using Compile Time polymorphism
/// </summary>
class MyBaseClass
{
    public void Show()
    {
        Console.WriteLine("Default message");
    }
    public void Show(string message)
    {
        Console.WriteLine(message);
    }
    public void Show(string message, string userName)
    {
        Console.WriteLine("The actual message: " + message + ",Username: " + userName);
    }
}
```

(parameter) string userName

The **Run time polymorphism** also named as the Overriding. This concept works in between classes or multiple classes or parent child classes where the child class has to get the behavior of the base class by inheriting the base class.

In this concept we generally have an abstract or virtual method in the base class and we override that method in the child class by using the override method.

e.g.

```
/// <summary>
/// Late Binding behaviour using Overriding
/// </summary>
abstract class MyBaseClass
{
    public abstract void Show();
    public abstract void Show(string message);
}

class ChildClass : MyBaseClass
{
    public override void Show()
    {
       Console.WriteLine("default message");
    }

    public override void Show(string message)
    {
        Console.WriteLine("The message is: " + message);
    }
}
```

So now we know the Compile Time polymorphism and Run Time polymorphism. The compile time polymorphism uses the concept of early binding and Run time polymorphism uses it as the late binding.

In early binding, the runtime (CLR) gets the behavior in the compilation of the program. It means that the method behavior will get compiled before in the early binding.

In Late binding, like Overriding, the behavior of the class and methods gets by the CLR when creating the object means at runtime. So, in the late binding the behavior of the class members identified by the CLR at runtime.

Now come to the next part of the question-which is good?

One can't say about the answer of this question, there are the scenarios where the early binding is good. When you have lot of objects and in that case, the early binding behavior performs well. While the late binding will be good when we have less objects. Let's say you want the dropdown list to be loaded when you click on it and not while the loading of the page. So in some scenario, it will be good if we have while load and it will not be good when you click.

So it's all depends on how you have implemented and the form structure.

3. In garbage collection, how the object generations come in the picture? How many generations an object can have? Please tell me the process of disposing the objects based on the generations? Can an object move from one generation to another? If yes then why? What's the need to have different generations as we are going to dispose the objects which are marked by the Garbage collector?

Ans.

Let's start with **what is Garbage collection** first and then we will come to our main question of the post. As we know that all the objects created using the **new** operator gets fits in to the Heap memory. So whenever a new object gets created, it tries to fit in the heap memory. Now let's say the heap memory is full and there is no place to keep another newly created object.

In that case the Garbage collector installed, which is the background process, runs through CLR and take the unused objects memory. It mainly cleanup the heap memory and the**new** objects get placed in to it.

Now the question comes that for which objects it reclaims for the memory?

How the object generations come in the picture?

It depends on the objects generations. The CLR finds out the objects which are no longer used by the application since longer time and then the Garbage collection reclaim their memory.

How many generations an object can have? Please tell me the process of disposing the objects based on the generations?

Actually there are 3 generations exists for the objects which are written under the .Net framework library. When a new object gets created, by default it moves to the generation 0.

Can an object move from one generation to another?

Now when the generation 0 objects gets occupied with the memory and garbage collector gets called by the run-time. It checks the objects which are no longer used by the application and mark them for deletion. After deleting or reclaim the memory, the older objects moved to next generation i.e. Generation 1. Now the next time the CLR will check for the Generation 1 object too and if it finds that in generation 1 if the objects are not used since longer time, it will mark them for release and move the remaining objects to generation 2.

In generation the objects which are under the main method, exists as they gets removed either at the end of the program or when both the generation 0 and generation 1 objects are using.

What's the need to have different generations as we are going to dispose the objects which are marked by the Garbage collector?

With the different generation, it improves the performance of the application as the Garbage collector need not to check for each of the objects in the memory. It first checks for the generation 0 objects and reclaim the memory. If still needs then goes to the generation 1 and then 2.

4. What is object graph in garbage collector? Is this the graph physically exists? Or how this graph gets created?

Ans. When the Garbage Collector gets called by the CLR to DE-allocate the memory in the heap, the Garbage Collector start finding the references of all the reachable objects which are currently in use. So it find the objects which are used by the processes and for rest of objects which are un-reachable or the Garbage collector is not able to find the references for them, it marks them for deletion.

Here the Garbage collector makes an Object graph which keeps track of all the objects which are marked for deletion. After the deleting the references for those objects, the heap memory gets compacted and a new root becomes available to use by the new created object.

Is this the graph physically exists? Or how this graph gets created?

No, this object graph creates virtually by the Garbage Collector to keep all the objects and to make them for deletion. This is the Garbage Collector responsibility to create this object graph and gets the references of each reachable object which are used by the application

5. Can we suppress the Garbage collector? If yes, then why do we need to suppress it as it is used to reclaim the unused memory and which improve s the performance of our application?

Ans. Yes, We can suppress the Garbage Collector. There is the Static method in GC class called as SupressFinalize.

```
GC.SuppressFinalize(objectName);
```

This Static method takes a parameter for the object. So we can pass it to suppress the claiming memory for this object.

Now the question comes "why do we need to suppress it as it is used to reclaim the unused memory",

So, whenever we are using dispose method for class object,which is capable of

disposing the object and in that case we don't need to use this method to again reclaim the memory.

e.g.

```
public class DemoClass : IDisposable
{
  public void Dispose()
  {
     Dispose(true);
     GC.SuppressFinalize(this);
  }
}
```

As we have seen above,the DemoClass is inherited by IDisposable interface and which have the Dispose method to implement.

Hence after implementation of Dispose() method, we need not to reclaim the memory using the Garbage collector and so we can use the SuppressFinalize() for the current class object.

6. We already have the Finalize method which reclaims the memory using the Garbage collector. This is automatic process then why do we have the Dispose () method? Is this method different from Finalize ()? What is the interface from which the Dispose () method inherited?

Ans. Yes, We have the Finalize() method which is used to reclaim the memory for the unused objects. The Finalize() method is sufficient to release the memory from heap for the un-referenced object but it is only for the managed objects. So Finalize() method can reclaim the managed objects memory from the heap which are not used since longer time.

Then what about the objects which are unmanaged? The objects which are out of .Net CLR? The objects where the CLR can't be used to manage? Dispose() method is used for all those objects which do not comes under CLR or under the Managed heap. Dispose() method can be overrides and can be written to reclaim the object of those classes. Dispose() method is implemented by using the IDisposable interface.

e.g.

```
public class TestClass : IDisposable
{
    public void Dispose()
   {
     Dispose(true);
   }

}
```

7. Can we call the Finalize() method to be executed explicitly when we want that particular object memory to be reclaim by the Garbage Collector? If yes, then where do we need to write the code? If no then why?

Ans. No, the Finalize() method can't be called as Explicitly. The Garbage collector calls it when the Heap memory is having no space to allocate the new object in the heap.

Finalize() method is mainly used to perform the cleanup on the unmanaged resources which are held by the current object.

Working Finalization: Until the Finalize method is overrides, the Garbage collection doesn't mark the object to claim its memory.Internally the Finalization Queue keep each of the objects which are marked for deletion by GC and then the Finalize()

method gets called.

The Object must be inaccessible(no reference found) to put or keep in the Finalization Queue.

Limitations: There are the limitation for Finalization:

a. Un-Deterministic - it is hard to find that when the Finalize() method will be called.

b. There is no guarantee that which object will be disposed first.

The C# compiler doesn't allow the overriding of Finalize() method an due to that we need to write the Destructor of the class.

As the GC is non-Deterministic so we can implement the Dispose() method explicitly to claim the unused memory by the Garbage collection.

For More details on GC, check the link:

GC Internal

8. Can we inherit child class from 2 base classes? If yes then how? If not then why? What is this scenario called as in OOPs? How to implement this kind of scenario where we need to inherit the methods from more than one base class?

Ans. No, We can't inherit more than one base class in to child class. This leads to the Multiple Inheritance where the child class can inherit more than one base class and the Multiple Inheritance doesn't support in .Net.

So this scenario is called as Multiple inheritance(a type of inheritance in OOPs).

If you do so, you will see the error 'Interface name expected.' as below:

```
class A
{
    // some implementation
    public void Show()
    {
        // do something
    }
}

class B
{
    // some implementation
    public void Print()
    {
        // do something
    }
}

class C : A, B
{
}
```

'B': interface name expected

To implement these kind of scenarios, we need the help of Interfaces, Where we can convert the second base class to Interface and then we can inherit one base class and second Interface.

```
class A
{
    // some implementation
    public void Show()
    {
        // do something
    }
}

interface B
{
    // some declaration
    void Print();
}

class C : A, B
{
    // interface method implementation
    public void Print()
    {
        // some implementation
    }
}
```

9. How the Virtual methods are different from General methods? Can we have a method in the base class and then in the child class can we write the same method? If not? Why? What is the error we will get if we write the same method in the child class with the same name as the parent class method name?

Ans. Yes, Virtual methods are different from the general method. The Virtual methods are the special type of methods which can be overrides in the child class and then the child class method will be the output or the default result.

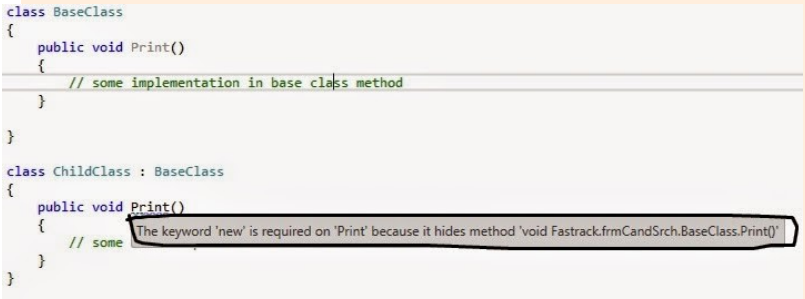
```
class BaseClass
{
    public virtual void Print()
    {
        // some implementation in base class method
    }

}

class ChildClass : BaseClass
{
    public override void Print()
    {
        // some other implementation in overriden method
    }
}
```

And when creating the child class object, it will take the child class method as the preference as below:

```
ChildClass c = new ChildClass();
c.Print();
```

Now, if we have the same method in the child class as base class then it will hide the base class method by default and what ever is implemented in the child class will

get executed.



So here we can see that the error will not come but it will show the warning to use the 'new' operator instead to hide the base class method. This scenario is called as **Shadowing** in the OOPs.

10. Why do we need abstract class or abstract members? Can't we simply write the general methods and fulfill our requirements? Can we get any advantage by implantation of abstract members? As per the abstract class scenario, let's say we have an abstract method called Show () in the abstract base class. Now if I am inheriting this base class to a child class, we need to override this abstract method to the child class. And then we will call this method by creating instance of child class.Now if we only have the method in the child class and then create the instance and call the same method? Then why abstract class method? Is that method doing anything here?
    Ans. Abstract class or abstract members(methods, properties) are those which are not complete. So when the class contains at-least one abstract member,the class must be defined with abstract class.
    We can also say that 'When the class does not provides the full functionality, we can declare the class as abstract'. the abstract class works based on the child class memory. Due to that we cant create the object of Abstract class, instead we can create the reference of the abstract class.

11. How abstract class and interfaces are different? Can't we create an abstract class by having all the members as abstract members and wherever required inherit and implement its members? Then why interface? I think interface is also doing the same thing? Then how they are different?

12. Can we have abstract properties in Interface? If yes, then how to write the syntax for the abstract property which is having the return type string?

13. Can we write static methods inside a non-static class? Is it possible to call a non-static method inside the static method? If yes then how?

14. Shadowing is the special type of overriding? How? Please explain?

15. When we inherit a class which is having the private members. Are the private members also gets inherited? If yes? Why cant' we get them by the class object? If no then why?

16. What is the Difference between Coupling and Cohesion? If the components are more cohesive the software is good? Or vice versa?

17. See the below code snippet and think about the output.

```
using System;

namespace ConsoleApplicationEx
{
  class Program
  {
    static void Main(string[] args)
    {
      string s = "string";
      Invoke(s);
      Console.ReadLine();
    }
    static void Invoke(object s)
    {
      Console.WriteLine("Object param invoked");
    }
    static void Invoke<T>(params T[] values)
    {
      Console.WriteLine("Params method invoked");
    }
  }
}
```

18. What is the Difference between HTTP enabled WCF Service and Web Services? I think if we restrict the WCF service just for the HTTP communication, both WCF and Web Service will have the same behavior. Then why still people prefer to have the WCF service rather than the Web Service. What all the things which can be achieved through HTTP enabled WCF service and cant be just from Web Service.

19. What is the Difference between SOAP enabled Services and ReST Services? Which one is preferred and why?

19 Brainstorming questions with answers in Microsoft Technologies->.Net, C#, ASP.Net, SQL Server 2008, ASP.Net MVC 3

This article contains few questions related to the .Net Technologies. it covers few question related to OOPs concepts, C# and ASP.Net, MVC with SQL Server. These questions are having the answers with them so that they will be helpful for those who are preparing for the interviews. These are the questions which are asked in various interviews in different companies. So in that way also it

will be useful as per the current industry trend and demands for 2+ years experience guys in .Net Technologies.

1. Why we used Inheritance. What is the benefit of this.
Ans. Inheritance is a way to inherit the base class members to the child class so that we can save lot of memory.
let's say, if you are already having some properties defined in the base class and then if you are inheriting this base class to the child class, the extra memory need not to be wasted to declare the base class members again. While inheriting, automatically they will be inherited.
e.g.

```
class Mybase
{
 int i, j;
 float z;
}
class MyChild: Mybase
{
 // no need to declare again the above variables
 // as they will be automatically inherited.
}
```

As we need not to declare the variables again in the child class, so we have saved here 4 + 4 + 8 = 16 bytes of memory for the child class.
Main usage of inheritance is 'Re-usability'. No need to declare variables, methods again.. If u want, then use them from base class. It reduce your code from complexity.

2. Why C# does not support Multiple Inheritance.
Ans. Multiple Inheritance is the situation in which when there are 2 base class and a single child class is trying to inherit the members from both of them then there is the confusion that which base class member should be inherited?
e.g.

```
class MyClass1
{
 public void Show()
 {
  // something
 }
}
class MyClass2
{
 public void Show()
 {
  // something else
 }
}
class MyChildClass: MyClass1, MyClass2
{
 // not possible
}
```

Due to this, the architecture of .Net or the framework doesn't support such situation and will throw the compile time error.
In C++, it is possible because there both the methods in the different base classes will have the different memory location and object will be accessed through the memory location. But in Java or .Net the object accessed through the class members objects.
It is quite difficult to implement multiple inheritance in C#. But we can do this through Interfaces.

3. Why we used virtual keyword.
Ans. If we want to override the members of the base class, we can make them as virtual. The virtual members have the capability to override them in the child class. If they are not overridden, the base class implementation will be executed.
It is not the mandatory to override the virtual class members.
e.g.

```
class MybaseClass
{
 public virtual void Show()
 {
  Console.WriteLine("hey..I am in base class");
 }
}

class MyChild: MyBaseClass
{
 public override void Show()
 {
  Console.WriteLine("hey..I am in child class");
 }
}
```

The preference will be given to child class always.

```
public static void Main(string[] args)
{
 MyChild objChild = new MyChild();
 objChild.Show(); // child class method will be called.
}
```

If a base class method needs to be overridden, it is defined using the keyword virtual (otherwise the sealed keyword is used to prevent overriding).

4. If my base class does not have virtual keyword but both class have same method , parameter and return type. then what will happen. base class method will be called or only child class method will called.
Ans. By default the preference will be given to child class. The child class method will be called always. If you want to call the base class method, then you need to use the base keyword like `base.MethodName()`

5. What are events and actions in c#.
Ans. Events are the objects which are binded with the particular action. Whenever an action happens, an event fires. Like button click event.The action is click and event is Button_Click which is attached with the event handler to do the particular action and get the result.

6. What is user define function in SQL or What are the user define function available in SQL.

Ans. There are 2 types of functions in SQL Server-

1. Inline functions

2. Tabular functions

Inline functions gives the result as a single value while the tabular functions return the multiple values like a table.

Both of these types of function can be created as user defined. Means we can create our own function which can be inline or Tabular by using their syntax.

```
Create function
return int[/Table]
as
Select FirstName+ LastName from Emp where EmpId = 101;
Create Function
return Table
as
Select * from Emp;
```

7. If we have one Master Page , one aspx page and one ascx page then which one will called 1st while page_load event and which one called 1st while page_Unload event.

Ans. If the content page contains the user controls then below will be the calling route-

Master Page Load event --> Content Page Load event --> User control page Load event

While unloading:

Content page Unload --> user control unload --> Master page unload event

8. What is Indexers and please give the description with example.

Ans. Indexer are the objects which doesn't need to be initialized and we can use them as it is for the storing and retrieving the data in to pages.

Indexer is represented as []. So we can use them in Session, Application types of objects when keeping the state management data.

e.g.

```
ViewState["UserName"] = txtUserName.Text;
Session["UserName"] = txtUserName.Text;
Application["UserName"] = txtUserName.Text;
```

To retrieve these data, we can simply type cast them as :

```
lblUserName.Text = ViewState["UserName"] as string;
lblUserName.Text = Session["UserName"] as string;
lblUserName.Text = Application["UserName"] as string;
```

Here you can see that there is no need to create object of the indexers. Directly we can retrieve the data.

9. What are indexes in Database. How many types of Indexes in SQL Server

Ans. Indexing is used to increase the performance of a query, instead of performing the entire table scan the query uses the index to execute the query.

There are five types of indexes in the SQL Server database:

1. Clustered Index

2. Non-Clustered Index
3. KeySet Index
4. Default Insex
5. XML Index
One thing here to remember that the Indexes retrieves the data by using the Binary tree formats.**1. Clustered Index:-** Those indexes which can be maximum 1 per table and they will be the key to retrieve the data. the clustered indexes retrieves the data in the B-Tree format and improve the performance of the retrieval from the table.
e.g. There are 1000 records in the table and the user want to retrieve the 234th record. Without index, the table scan will compare the record in whole table to get the matched record. So for the 1000 records, it will match for 999 times. If it found the records in the beginning, then also it will search the whole table.
When using the index, the below process will occur for the searching of 234th record:
1. It will take the 234th record as the root and compare the table by partitioning it in to two parts.1-500 and 501-1000
2. then it compares with the left and right and then it will found that 234 is less than 500 so leave the right part.
3. Now comparing will happen for the 1-250 and then 250-500. For this, it will check that 234 will come in the 1-250 range so leave the other part.
4. Now the comparison will be 1-125 and 126-250...and so on..
So if you calculate the total comparison it will be hardly 8-9. So we have reduces almost 990 comparison to improve the performance.
In Clustered index, the actual data exists at the leaf root of the B-tree and it search the data directly.**2. Non-Clustered Index:-** it searches the data in B+ tree format where the leaf node contains the memory location of the data. Once it gets the actual memory location, by using the reference it retrieve the actual data.There can be more than one Non-Clustered indexes per table. There can be 249 non-clustered index(in SQL Server 2005) and 999 non-clustered index (SQL Server 2008).**3. KeySet Indexes:-** Only keys are stored and not the actual data. the search will happens based on the key. Once the key found, it searches the complete data for the key. So its faster than all the indexes.**4. Default Index:-** When the primary is added to the table, by default the index gets created called as Default Index.**5. XML index:-** When the index is created on the XML column, the index is called as the XML index. This type of index gets introduces since SQL Server 2005 version. XML type column is supported since SQL Server 2005.

10.Why we use Interface.
Ans. Interface is a way to use the global functionality throughout the application. If your application require many unrelated object types to provide certain functionality then you go for interfaces.
It defines a contract between the application and the object.
Let's suppose we want to Show the data or Print the data in many pages of the application, we can create a Interface which will contain the Abstract method and in the page where we want, we can implement it as per our requirements.
e.g.

```
interface inf
{
 void Print(); // abstract method
}
internal class MyClass: inf
```

```
{
 public void Print()
 {
  // functionality to print to PDF document
 }
}
internal class MyTest: inf
{
 public void Print()
 {
  // functionality to print to tiff document
 }
}
internal Class MyTest: inf
{
 public void Print()
 {
  // functionality to print to .jpg format
 }
}
```

We can see here we have separate classes and the implementation of the Print method is different without making any changes in our interface.
So if we want to implement something global, we can use the Interface, for local or limited to class, we can use the abstract class.

11.Why we use Method Overloading

Ans. To reduce the memory and the good readability, we use the Overloading. This is the concept where we can have the same method name for similar work with different behavior.
Let's say we want to get the Database connection based on the provider name to connect with different databases, we can use the overloaded methods like:

```
public string GetConnectionString()
{
 // return default connection string
}
public string GetConnectionString(string provider)
{
 // return connection string based on provider name
}
```

12. Why we use Method Overriding

Ans. Method overriding is the concept where we can use the same method in parent as well as in child class to reduce the memory and use the similar behavior. If we don't want to change the behavior, we need not to override it. So to make the changes in the behavior, we use the overriding.

13. What is the difference between Abstraction and Encapsulation.

Ans. Encapsulation is the biding and hiding of the data while Abstraction is to get the essential

information from the raw data.
e.g. Encapsulation

```
class Test
{
 private string _name;
 public string Name
 {
  set _name= value;
  return _name;
 }
}
```

Here the _name is the private which is not accessible so hiding the data. All the data will be binded to the private variable as per the statement

```
set _name= value;
```

This property will be accessed by the public member called Name.
So Binding and hiding means encapsulation.
Abstraction is done by using the abstract class where we can have abstract and non abstract members.
When ever we are having the abstract members, we need to implement it/override them in the child class to use them.

```
class MyBaseClass
{
 public abstract void Displaye();
}
class MyChildClass: MyBaseClass
{
 public override void Display()
 {
  // do something
 }
}
```

Here the essential things is the abstract method which must be overridden. If we add any number of non-abstract method, the child class wont care. So Getting the essential information is Abstraction.

14. In which case we can use Abstract class in our project.
Ans. Abstract class is always used when we have the limited scope or the objects are of similar types.
e.g. Lets suppose we want to get the area of few objects. This is not the global where we need to do.
So we can create the abstract method to calculate the area and then according to shape, we can override them. So here the scope is limited and will not be used throughout the application.
Hence we can use the abstract class.

15. What is the difference between Array List and Ilist.
Ans. IList is an interface for the implementing the List collection.
As the List is the generic so we can use it to restrict the type of the list e g.

```
List objList = new List();
```

It means, the List object is restricted to the int. We can't insert here any type of data in to list. Only the int is permitted.
ArrayList is the collection where we can insert any type of data as:

```
ArrayList objArrayList = new ArrayList();
objArrayList.Add("Hello");
objArrayList.Add(1);
objArrayList.Add("Sure");
```

The ArrayList will not check the type in the compilation of the program.

16. What is difference between internal class and sealed class.

Ans. The internal class object can be used with-in the assembly and not outside of the assembly. These classes can be inherited.
Sealed classes are those classes which doesn't need anything from outside. These are full classes. These classes can't be extended.

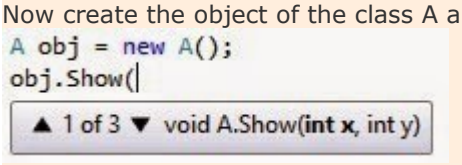
17. Are these method are overloading methods?

```
public void show(int x , string y)
public void show(string x , int y)
```

Ans. Yes, these are the overloading methods because, their signatures are not same
Signature= Method Name + Argument types

```
class A
{
    public void Show(int x, int y) { }
    public void Show(string x, string y) { }
    public void Show(int x, string y) { }
}
```

Both signatures are different. So overloading methods.
Now create the object of the class A and check for the overloaded method:



As per method overloading, the method should differ in
- type of parameters
- Order of parameters
- number of parameters

18. What is the difference between Remoting and web service.

Ans. Remoting is the concept of calling the remote interfaces like communication with the heterogeneous applications but both should be developed using the same .Net language.
For the client and server communication, we use the Remoting.
Remoting is the platform dependent so both the client and server must be built in .Net Technology and both the systems should use the CLR.
Web-service is small logic which runs on the internet. It is used for the heterogeneous applications and consume the service in any application. It is platform independent so no need to have the CLR or .net framework to be installed on the machines. C++ web service can be consumes in .Net application.

19. Can we use viewstate in MVC.

Ans. No, ViewState cannot be used in MVC. There is no concept of ViewState in the MVC as it is not required. The MVC views are not having anything server side code so the view only be

rendered and not loaded.
To pass the data from one page to another, we can use the ViewBag.

The reason we use MVC is to separate business logic from your view or UI and the site should be easily testable.
ViewState mixes your business logic with your UI, while MVC separates the Business Logic from the UI.
Hope they will be useful to all of us.