1 :: What is Abstract method?

Abstract method doesn't provide the implementation & forces the derived class to override the method.

2 :: What is Virtual method?

Virtual Method has implementation & provide the derived class with the option to override it.

3 :: What is Polymorphisms?

Polymorphism means one interface and many forms. Polymorphism is a characteristics of being able to assign a different meaning or usage to something in different contexts specifically to allow an entity such as a variable, a function or an object to have more than one form.

There are two types of Polymorphism.
Compile time: function or operator overloading
Runtime: Inheritence & virtual functions

4 :: What is Abstract Class?

Abstract class is a class that can not be instantiated, it exists extensively for inheritance and it must be inherited. There are scenarios in which it is useful to define classes that is not intended to instantiate; because such classes normally are used as base-classes in inheritance hierarchies, we call such classes abstract classes.

Abstract classes cannot be used to instantiate objects; because abstract classes are incomplete, it may contain only definition of the properties or methods and derived classes that inherit this implements it's properties or methods.

Static, Value Types & interface doesn't support abstract modifiers. Static members cannot be abstract. Classes with abstract member must also be abstract.

5 :: When to use Interface over abstract class?

Abstract Classes: Classes which cannot be instantiated. This means one cannot make a object of this class or in other way cannot create object by saying ClassAbs abs = new ClassAbs(); where ClassAbs is abstract class.
Abstract classes contains have one or more abstarct methods, ie method body only no implementation.
Interfaces: These are same as abstract classes only difference is we can only define method definition and no implementation.
When to use wot depends on various reasons. One being design choice.
One reason for using abstarct classes is we can code common
functionality and force our developer to use it. I can have a complete

class but I can still mark the class as abstract.
Developing by interface helps in object based communication.

6 :: What is Sealed modifiers?

Sealed types cannot be inherited & are concrete.
Sealed modifiers can also be applied to instance methods, properties, events & indexes. It can't be applied to static members.

Sealed members are allowed in sealed and non-sealed classes.

7 :: What is Inheritance?

It provides a convenient way to reuse existing fully tested code in different context thereby saving lot of coding.

Inheritance of classes in C# is always implementation Inheritance.

8 :: What is New modifiers?

The new modifiers hides a member of the base class. C# supports only hide by signature.

9 :: What is Virtual keyword?

This keyword indicates that a member can be overridden in a child class. It can be applied to methods, properties, indexes and events.

10 :: What is an Interface?

An interface is a contract & defines the requisite behavior of generalization of types.

An interface mandates a set of behavior, but not the implementation. Interface must be inherited. We can't create an instance of an interface.

An interface is an array of related function that must be implemented in derived type. Members of an interface are implicitly public & abstract.

An interface can inherit from another interface.

11 :: What is Static Method?

It is possible to declare a method as Static provided that they don't attempt to access any instance data or other instance methods.

12 :: What is Static field?

To indicate that a field should only be stored once no matter how many instance of the class we create.

13 :: What is Class?

A Class is the generic definition of what an object is a template.

The keyword class in C# indicates that we are going to define a new class (type of object)

14 :: What is Object?

Object is anything that is identifiable as a single material item.

15 :: Can Struct be inherited?

No, Struct can't be inherited as this is implicitly sealed.

16 :: What is Method Overriding? How to override a function in C#?

Use the override modifier to modify a method, a property, an indexer, or an event. An override method provides a new implementation of a member inherited from a base class. The method overridden by an override declaration is known as the overridden base method. The overridden base method must have the same signature as the override method.

You cannot override a non-virtual or static method. The overridden base method must be virtual, abstract, or override.

17 :: What is Method overloading?

Method overloading occurs when a class contains two methods with the same name, but different signatures.

18 :: What is Overriding?

Method overriding is a feature that allows to invoke functions (that have the same signatures) and that belong to different classes in the same hierarchy of inheritance using the base class reference. In C# it is done using keywords virtual and overrides .

19 :: Default Access modifiers in C#?

An enum has default modifier as public

A class has default modifiers as private . It can declare members (methods etc) with following access modifiers:
public

protected
internal
private
protected internal

An interface has default modifier as public

A struct has default modifier as private and it can declare its members (methods etc) with following access modifiers:
public
internal
private

20 :: Can we specify the access modifier for explicitly implemented interface method?

No, we can't specify the access modifier for the explicitly implemented interface method. By default its scope will be internal.

21 :: What is Protected Internal access modifier in C#?

Protected Internal is a access modifiers for the members (methods or functions) ie. you can't declare a class as protected internal explicitly. The members access is limited to the current assembly or types derived from the containing class.

Protected Internal means the method is accessible by anything that can access the protected method UNION with anything that can access the internal method.

22 :: What is Protected access modifier in C#?

The protected keyword is a member access modifier. It can only be used in a declaring a function or method not in the class ie. a class can't be declared as protected class.

A protected member is accessible from within the class in which it is declared, and from within any class derived from the class that declare this member. In other words access is limited to within the class definition and any class that inherits from the class

A protected member of a base class is accessible in a derived class only if the access takes place through the derived class type.

23 :: What is Internal access modifier in C#?

The internal keyword is an access modifier for types and type members ie. we can declare a class as internal or its member as internal. Internal members are accessible only within files in the same assembly (.dll). In other words, access is limited exclusively to classes defined within the current project assembly.

24 :: What is Private access modifier in C#?

The private keyword is a member access modifier ie. we can't explicitly declare a class as Private, however if do not specify any access modifier to the class, its scope will be assumed as Private. Private access is the least permissive access level of all access modifiers.

Private members are accessible only within the body of the class or the struct in which they are declared. This is the default access modifier for the class declaration.

25 :: What is Public access modifier in C#?

The public keyword is an access modifier for types and type members ie. we can declare a class or its member (functions or methods) as Public. There are no restrictions on accessing public members.

26 :: What is pure virtual function in OOP?

When you define only function prototype in a base class without and do the complete implementation in derived class. This base class is called abstract class and client won't able to instantiate an object using this base class.

A pure virtual function is a function that must be overridden in a derived class and need not be defined. A virtual function is declared to be "pure" using the curious "=0"
syntax:

```
class Base {
public:
void f1(); // not virtual
virtual void f2(); // virtual, not pure
virtual void f3() = 0; // pure virtual
};
```

27 :: When to Use Abstract Classes and When Interfaces.

If you anticipate creating multiple versions of your component, create an abstract class. Abstract classes provide a simple and easy way to version your components. By updating the base class, all inheriting classes are automatically updated with the change. Interfaces, on the other hand, cannot be changed once created. If a new version of an interface is required, you must create a whole new interface.

If the functionality you are creating will be useful across a wide range of disparate objects, use an interface. Abstract classes should be used primarily for objects that are closely related, whereas interfaces are best suited for providing common functionality to unrelated classes.

If you are designing small, concise bits of functionality, use interfaces. If you are designing large functional units, use an abstract class.

If you want to provide common, implemented functionality among all implementations of your component, use an abstract class. Abstract classes allow you to partially implement your class, whereas interfaces contain no implementation for any members.

28 :: What are Constructors?

Constructors are used for initializing the members of a class whenever an object is created with the default values for initialization.

If no constructor defined then the CLR will provide an implicit constructor which is called as Default Constructor.

A class can have any number of constructors provided they vary with the number of arguments that are passed, which is they should have different signatures.

Constructors do not return a value
Constructors can be overloaded

29 :: What are the various types of Constructor

Public : Accessible to All
Private: Those classes in which only static members are there and you don't want there objects to be created in any class.
Static: Used for initializing only the static members of the class. These will be invoked for the very first time the class is being loaded on the memory. They cannot accept any arguments. Static Constructors cannot have any access modifiers.
Intern: implementations of the abstract class to the assembly defining the class. A class containing an internal constructor cannot be instantiated outside of the assembly (Namespace). and External

30 :: Whats the Difference between Interface and Abstract Class?

Abstract Class:
Have constructors.
Not necessarily for the class inheriting it to Implement all the Methods.
Doesn't Support Multiple Inheritance.

Where everything is Opposite in the Interfaces.

31 :: What can you do to make class available for inheritance but you need to prevent its method to come in inheritance chain?

Well, Declare a class with public access specifier and mark all it's method to sealed . As anything which is declared with sealed keyword cannot be inherited.

32 :: What Are Attributes in DotNet?

An Attribute is a declarative tag which can be used to provide information to the compiler about the behaviour of the C# elements such as classes and assemblies.
C# provides convenient technique that will handle tasks such as performing compile time operations , changing the behaviour of a method at runtime or maybe even handle unmanaged code.
C# Provides many Built-in Attributes

Some Popular ones are

- Obsolete
- DllImport
- Conditional
- WebMethod

and Many more.
Members please keep on posting more responses providing more In-Built attributes.

Regards Hefin Dsouza

33 :: What is Polymorphism?

In OPP'S, polymorphism(Greek meaning "having multiple forms") is the ablity of being able to assign a different meaning or usage to something in different contexts - specifically, to allow an entity such as a a function, or an object to have more than one forms.

In C# :
Parent classes may define and implement "virtual" methods(Which is done using the "virtual" keyword), and derived classes can override them(using the "override" keyword), which means they provide their own definition and implementation.At run-time, when user's code calls the method, the CLR looks up the run-time type of the object, and invokes that override of the virtual method. Thus in your source code when a method of the base class is called it executes the overriden method.

34 :: Can we declare private class in a Namespace?

No. If you try to create a private class in a Namespace, Compiler will throw a compile time error "Namespace elements cannot be explicitly declared as private, protected, or protected internal".

Reason: The message says it all. Classes can only be declared as private, protected or protected internal when declared as nested classes, other than that, it doesn't make sense to declare a class with a visibility that makes it unusable, even in the same module. Top level classes cannot be private, they are "internal" by default, and you can just make them public to make them visible from outside your DLL.

35 :: What is a private constructor? Where will you use it?

When you declare a Constructor with Private access modifier then it is called Private Constructor. We can use the private constructor in singleton pattern.

If you declare a Constructor as private then it doesn't allow to create object for its derived class, i.e you loose inherent facility for that class.

Example:

```
Class A

{

// some code

Private Void A()

{

//Private Constructor

}

}


Class B:A

{

//code

}


B obj = new B();// will give Compilation Error
```

Because Class A constructor declared as private hence its accessibility limit is to that class only, Class B can't access. When we create an object for Class B that constructor will call constructor A but class B have no rights to access the Class A constructor hence we will get compilation error.

36 :: In which cases you use override and new base?

Use the new modifier to explicitly hide a member inherited from a base class. To hide an inherited member, declare it in the derived class using the same name, and modify it with the new modifier.

37 :: Can we call a base class method without creating instance?

Yep. But ..

► Its possible If its a static method.

► Its possible by inheriting from that class also.

► Its possible from derived classes using base keyword.