**13. What is the difference between a class and a structure?**

**Class**:

1. A class is a reference type.
2. While instantiating a class, CLR allocates memory for its instance in heap.
3. Classes support inheritance.
4. Variables of a class can be assigned as null.
5. Class can contain constructor/destructor.

**Structure**:

1. A structure is a value type.
2. In structure, memory is allocated on stack.
3. Structures do not support inheritance.
4. Structure members cannot have null values.
5. Structure does not require constructor/destructor and members can be initialiazed automatically.

**14. What are similarities between a class and a structure.**

Structures and classes are the two most important data structures that are used by programmers to build modular programs by using OOP languages, such as Visual Basic .NET, and Visual C#. The following are some of the similarities between a class and a structure:

* Access specifiers, such as public, private, and protected, are identically used in structures and classes to restrict the access of their data and methods outside their body.
* The access level for class members and struct members, including nested classes and structs, is private by default. Private nested types are not accessible from outside the containing type.
* Both can have constructors, methods, properties, fields, constants, enumerations, events, and event handlers.
* Both structures and classes can implement interfaces to use multiple-inheritance in code.
* Both structures and classes can have constructors with parameter.
* Both structures and classes can have delegates and events.

**28. State the features of an interface.**

An interface is a template that contains only the signature of methods. The signature of a method consists of the numbers of parameters, the type of parameter (value, reference, or output), and the order of parameters. An interface has no implementation on its own because it contains only the definition of methods without any method body. An interface is defined using the interface keyword. Moreover, you cannot instantiate an interface. The various features of an interface are as follows:

* An interface is used to implement multiple inheritance in code. This feature of an interface is quite different from that of abstract classes because a class cannot derive the features of more than one class but can easily implement multiple interfaces.
* It defines a specific set of methods and their arguments.
* Variables in interface must be declared as public, static, and final while methods must be public andabstract.
* A class implementing an interface must implement all of its methods.
* An interface can derive from more than one interface.

**36. What are abstract classes? What are the distinct characteristics of an abstract class?**

An abstract class is a class that cannot be instantiated and is always used as a base class.  
The following are the characteristics of an abstract class:

* You cannot instantiate an abstract class directly. This implies that you cannot create an object of the abstract class; it must be inherited.
* You can have abstract as well as non-abstract members in an abstract class.
* You must declare at least one abstract method in the abstract class.
* An abstract class is always public.
* An abstract class is declared using the abstract keyword.

The basic purpose of an abstract class is to provide a common definition of the base class that multiple derived classes can share.

**50. Differentiate between an abstract class and an interface.**

**Abstract Class**:

1. A class can extend only one abstract class
2. The members of abstract class can be private as well as protected.
3. Abstract classes should have subclasses
4. Any class can extend an abstract class.
5. Methods in abstract class can be abstract as well as concrete.
6. There can be a constructor for abstract class.
7. The class extending the abstract class may or may not implement any of its method.
8. An abstract class can implement methods.

**Interface**

1. A class can implement several interfaces
2. An interface can only have public members.
3. Interfaces must have implementations by classes
4. Only an interface can extend another interface.
5. All methods in an interface should be abstract
6. Interface does not have constructor.
7. All methods of interface need to be implemented by a class implementing that interface.
8. Interfaces cannot contain body of any of its method.