1. reference type vs. value type

1. value type will directly hold the value, while reference type will hold the memory address or reference for this value
2. value type is stored in stack memory, while reference types will be stored in heap memory
3. value type will not be collected by garbage collector, while reference type will be collected by garbage collector
4. value type can be created by Struct or Enum, but reference type can be created by class, interface, delegate, or array
5. value type cannot accept null values, but reference types can accept null values

2. boxing vs. unboxing

1. boxing: convert a value type to a reference type
2. unboxing: convert the reference type to a value type

- Example:
  - `int i = 10;`
  - `object o = i; //boxing`
  - `int j = (int) o; //unboxing`

3. abstract class vs. interface

1. Interface supports multiple inheritance, but abstract class does not
2. Interface cannot have instanced constructor, but abstract class can have
3. Interface has by default all members abstract and public, but abstract class can have abstract and concrete members
4. Interface cannot have fields, but abstract class can have fields.

4. Overriding vs. Overloading

1. Overloading - the ability to have multiple methods within the same class with the same name, but with different parameters.
2. Overriding - the ability to redefine the implementation of a method in a class that inherits from a parent class.

5. What does constructor do in a class? Can it be overridden? Can it be overloaded?

1. constructor is a special method which shares the same name of the class and doesn't have any return type, not even void
2. constructor is used to create an object of the class and initialize class members
3. if there is no constructor in the class, C# compiler will provide a default constructor
4. if we create any constructor ourselves, the default constructor will be replaced
5. constructor can be overloaded
6. constructor cannot be inherited so a constructor cannot be overridden
7. by default, the derived class constructor will make a call the base class constructor

6. What does static keyword do in C#?

It can declare a static member, which belongs to the type itself rather than to a specific object. The static modifier can't be used with indexers or finalizers.

7. Difference between Virtual method and Abstract method?

1. abstract method - do not provide an implementation and FORCE the derived classes to override this method
2. virtual method - can have an implementation and provide the derived classes with the OPTION of overriding it

8. what are delegates in C#, what are different types of built-in delegates

Delegates is a type of safe function pointer. There are three types: Func, Action, Predicate

9. Explain different access modifiers in C#

public: member can be accessed anywhere
protected: member can be accessed in the current class and its child classes
internal: member can be accessed in the current assembly
private: member can only be accessed in the current class

10. What is the extension method in C#? examples of built-in extension methods? How to create custom extension methods?

- Extension method - a way to add new functionality into an existing type (both reference and value)
- Example - LinQ
- How to create custom extension methods:
  - class containing extension methods must be a static class
  - method itself must be static
  - first parameter of extension method must be of the type which will be extended
  - first parameter must be written after 'this' keyword

11. Ref vs. Out vs. Params

- out mode: return more than one values -- use out keyword
- pass by reference: the actual value is passed to the formal parameters so any change in formal parameters will also reflect in actual parameters -- use ref keyword
- optional parameters: default values will be assigned to the optional parameter

12. Pass by reference vs. Pass by Value

1. pass by value: a copy of the actual parameter is created and will be passed into the formal parameters -- default
2. pass by reference: the actual value is passed to the formal parameters so any change in formal parameters will also reflect in actual parameters -- use ref keyword

13. array vs. arrayList

1. Array - strongly-typed collections of the same data type and have a fixed length that cannot be changed during runtime.
2. Array list - not a strongly-typed collection. It can store the values of different data types or same datatype.

14. example of encapsulation, where to implement

```
public class Example

{

      public int Id {set; get;};

      public string FullName {get; set;};

}
```

15. how do you handle exceptions? Syntax.

1. try(){codes to try}
2. catch(SomeSpecificException ex){//Code to handle the exception }
3. finally {//Code to execute after the try (and possibly catch) blocks}

16. what is generic, syntax to define

In C#, generic means not specific to a particular data type.

A generic type is declared by specifying a type parameter in an angle brackets after a type name, e.g. TypeName<T> where T is a type parameter.

17. what is LINQ

LINQ (Language Integrated Query) is uniform query syntax in C# to save and retrieve data from different sources.

18. IEnumerable vs. IQuerable

1. IEnumerable - when linq is working with in-memeory data source (list, array…)
2. IQuerable - when linq is working with out-of-memory data source

19. First vs. FirstOrDefault vs. Single vs. SingleOrDefault

1. First - return the first record when there is one or more records; if not matched --> throw an exception
2. FirstOrDefault - return the first record where there is one or more records, if not matched --> assign default value
3. Single - return the matched single record; if not matched --> throw and exception, if more than one match --> throw an exception
4. SingleOrDefault - return the matched single record; if not matched --> assign the default value, if more than one match --> throw an exception

20. Any vs. All

1. any() - check if any of the element satisfy the specific condition, if yes, return true, if no return false
2. all() - check if all the elements satisfy the specific condition, if yes, return true, if no, return false

21. Skip vs. Take

1. Take() method extracts the first n elements (where n is a parameter to the method) from the beginning of the target sequence and returns a new sequence containing only the elements taken.
2. Skip() operator bypasses a specified number of contiguous rows from a sequence/table and returns the remaining table. It can skip rows from the top or can be for a certain criterion, in other words it can also skip rows depending on a certain criterion. It works like NOT IN in SQL.

22. Deferred execution and Immediate execution in LINQ

1. Deferred Execution
    a. A query variable only stores the query commands. The actual execution of the query is deferred until you iterate over the query variable in a for each statement.
    b. This concept is considered as deferred execution.
    c. Deferred execution can greatly improve performance when you must manipulate large data collections.
    d. The collection results will have smaller memory footprints.
2. Forcing Immediate Execution
    a. Queries that perform aggregation functions over a range of source elements must first iterate over those elements.
    b. Examples of such queries are Count, Max, Average, and First. These execute without an explicit foreach statement because the query itself must use foreach in order to return a result.
    c. To force immediate execution of any query and cache its results, you can call the ToList<TSource> or ToArray<TSource> methods.