**Q #1) What is an Object and a Class?**

**Ans:**A Class is an encapsulation of properties and methods that are used to represent a real-time entity. It is a data structure that brings all the instances together in a single unit.

An Object in an instance of a Class. Technically, it is just a block of memory allocated that can be stored in the form of Variables, Array or a Collection.

**Q #2) What are the fundamental OOP concepts?**

**Ans: The four fundamental concepts of Object Oriented Programming are:**

* **Encapsulation** – The Internal representation of an object is hidden from the view outside object’s definition. Only the required information can be accessed whereas the rest of the data implementation is hidden.
* **Abstraction** – It is a process of identifying the critical behavior and data of an object and eliminating the irrelevant details.
* **Inheritance** – It is the ability to create new classes from another class. It is done by accessing, modifying and extending the behavior of objects in the parent class.
* **Polymorphism** – The name means, one name, many forms. It is achieved by having multiple methods with the same name but different implementations.

**Q #3) What is Managed and Unmanaged code?**

**Ans:** **Managed code** is a code which is executed by CLR (Common Language Runtime) i.e all application code based on .Net Platform. It is considered as managed because of the .Net framework which internally uses the garbage collector to clear up the unused memory.

**Unmanaged code** is any code that is executed by application runtime of any other framework apart from .Net. The application runtime will take care of memory, security and other performance operations.

**Q #4) What is an Interface?**

**Ans:** **An Interface** is a class with no implementation. The only thing that it contains is the declaration of methods, properties, and events.

**Q #5) What are the different types of classes in C#?**

**Ans: The different types of class in C# are:**

* **Partial class** – Allows its members to be divided or shared with multiple .cs files. It is denoted by the keyword *Partial.*
* **Sealed class** – It is a class which cannot be inherited. To access the members of a sealed class, we need to create the object of the class.  It is denoted by the keyword *Sealed*.
* **Abstract class** – It is a class whose object cannot be instantiated. The class can only be inherited. It should contain at least one method.  It is denoted by the keyword *abstract.*
* **Static class** – It is a class which does not allow inheritance. The members of the class are also static.  It is denoted by the keyword *static*. This keyword tells the compiler to check for any accidental instances of the static class.

**Q #6) Explain Code compilation in C#.**

**Ans: There are four steps in code compilation which include:**

* Compiling the source code into Managed code by C# compiler.
* Combining the newly created code into assemblies.
* Loading the Common Language Runtime(CLR).
* Executing the assembly by CLR.

**Q #7) What are the differences between a Class and a Struct?**

**Ans: Given below are the differences between a Class and a Struct:**

| **Class** | **Struct** |
| --- | --- |
| Supports Inheritance | Does not support Inheritance |
| Class is Pass by reference (reference type) | Struct is Pass by Copy (Value type) |
| Members are private by default | Members are public by default |
| Good for larger complex objects | Good for Small isolated models |
| Can use waste collector for memory management | Cannot use Garbage collector and hence no Memory management |

**Q #8) What is the difference between Virtual method and Abstract method?**

**Ans:**A **Virtual method** must always have a default implementation. However, it can be overridden in the derived class, though not mandatory. It can be overridden using *override* keyword.

An **Abstract method** does not have an implementation. It resides in the abstract class. It is mandatory that the derived class implements the abstract method. An *override* keyword is not necessary here though it can be used.

**Q #9) Explain Namespaces in C#.**

**Ans:** They are used to organize large code projects. “System” is the most widely used namespace in C#. We can create our own namespace and use one namespace in another, which are called Nested Namespaces.

They are denoted by the keyword “namespace”.

**Q #10) What is “using” statement in C#?**

**Ans:** “Using” Keyword denotes that the particular namespace is being used by the program.

**For Example,** *using System*. Here *System* is a namespace. The class Console is defined under System.  So we can use the console.writeline (“….”) or readline in our program.

**Q #11) Explain Abstraction.**

**Ans: Abstraction** is one of the OOP concepts. It is used to display only the essential features of the class and hides the unnecessary information.

**Let us take an Example of a Car:**

A driver of the car should know the details about the Car such as color, name, mirror, steering, gear, brake, etc. What he doesn’t have to know is an Internal engine, Exhaust system.

So, Abstraction helps in knowing what is necessary and hiding the internal details from the outside world. Hiding of the internal information can be achieved by declaring such parameters as Private using *the private* keyword.

**Q #12) Explain Polymorphism?**

**Ans:** Programmatically, **Polymorphism** means same method but different implementations.

It is of 2 types, Compile-time and Runtime.

**Compile time polymorphism** is achieved by operator overloading.

**Runtime polymorphism** is achieved by overriding. Inheritance and Virtual functions are used during Runtime Polymorphism.

**For Example**, If a class has a method Void Add(), polymorphism is achieved by Overloading the method, that is, void Add(int a, int b), void Add(int add) are all overloaded methods.

**Q #13) How is Exception Handling implemented in C#?**

**Ans: Exception handling is done using four keywords in C#:**

* **try** – Contains a block of code for which an exception will be checked.
* **catch** – It is a program that catches an exception with the help of exception handler.
* **finally** – It is a block of code written to execute regardless whether an exception is caught or not.
* **Throw** – Throws an exception when a problem occurs.

**Q #14) What are C# I/O Classes? What are the commonly used I/O Classes?**

**Ans:**C# has System.IO namespace, consisting of classes that are used to perform various operations on files like creating, deleting, opening, closing etc.

**Some commonly used I/O classes are:**

* **File** – Helps in manipulating a file.
* **StreamWriter** – Used for writing characters to a stream.
* **StreamReader** – Used for reading characters to a stream.
* **StringWriter** – Used for reading a string buffer.
* **StringReader** – Used for writing a string buffer.
* **Path** – Used for performing operations related to path information.

**Q #16) What is a Destructor in C#?**

**Ans:**A **Destructor** is used to clean up the memory and free the resources. But in C# this is done by the garbage collector on its own. System.GC.Collect() is called internally for cleaning up. But sometimes it may be necessary to implement destructors manually.

**For Example:**

~Car()

{

Console.writeline(“….”);

}

**Q #17) What is an Abstract Class?**

**Ans:** An **Abstract class** is a class which is denoted by abstract keyword and can be used only as a Base class. An Abstract class should always be inherited. An instance of the class itself cannot be created. If we do not want any program to create an object of a class, then such classes can be made abstract.

Any method in the abstract class does not have implementations in the same class. But they must be implemented in the child class.

**For Example:**

abstract class AB1

{

Public void Add();

}

Class childClass : AB1

{

childClass cs = new childClass ();

int Sum = cs.Add();

}

All the methods in an abstract class are implicitly virtual methods. Hence virtual keyword should not be used with any methods in  abstract class.

**Q #18) What are Boxing and Unboxing?**

**Ans:** Converting a value type to reference type is called **Boxing**.

**For Example:**

int Value1 -= 10;

//————Boxing——————//

object boxedValue = Value1;

Explicit conversion of same reference type (created by boxing) back to value type is called **Unboxing**.

**For Example:**

//————UnBoxing——————//

int UnBoxing = int (boxedValue);

**Q #19) What is the difference between Continue and Break Statement?**

**Ans:** Break statement breaks the loop. It makes the control of the program to exit the loop. **Continue statement** makes the control of the program to exit only the current iteration. It does not break the loop.

**Q #20) What is the difference between finally and finalize block?**

**Ans:*finally* block** is called after the execution of try and catch block. It is used for exception handling. Regardless of whether an exception is caught or not, this block of code will be executed. Usually, this block will have clean-up code.

finalize method is called just before garbage collection. It is used to perform clean up operations of Unmanaged code. It is automatically called when a given instance is not subsequently called.

Questions on Arrays and Strings

**Q #21) What is an Array? Give the syntax for a single and multi-dimensional array?**

**Ans: An Array** is used to store multiple variables of the same type. It is a collection of variables stored in a contiguous memory location.

**For Example:**

double numbers = new double[10];

int[] score = new int[4] {25,24,23,25};

A Single dimensional array is a linear array where the variables are stored in a single row. Above **example** is a Single dimensional array.

Arrays can have more than one dimension. Multidimensional arrays are also called rectangular arrays.

**For Example**, int[,] numbers = new int[3,2] { {1,2} ,{2,3},{3,4} };

**Q #22) What is a Jagged Array?**

**Ans:** A Jagged array is an array whose elements are arrays. It is also called as the array of arrays. It can be either single or multiple dimensions.

int[] jaggedArray = new int[4][];

**Q #23) Name some properties of Array.**

**Ans: Properties of an Array include:**

* **Length** – Gets the total number of elements in an array.
* **IsFixedSize** – Tells whether the array is fixed in size or not.
* **IsReadOnly** – Tells whether the array is read-only or not.

**Q #24) What is an Array Class?**

**Ans:** An **Array class** is the base class for all arrays. It provides many properties and methods. It is present in the namespace System.

**Q #25) What is a String? What are the properties of a String Class?**

**Ans:**A **String** is a collection of char objects. We can also declare string variables in c#.

string name = “C# Questions”;

A string class in C# represents a string.

The properties of String class are **Chars and Length.**  
**Chars** get the Char object in the current String.  
**Length** gets the number of objects in the current String.

**Q #26) What is an Escape Sequence? Name some String escape sequences in C#.**

**Ans:** An **Escape sequence** is denoted by a backslash (\). The backslash indicates that the character that follows it should be interpreted literally or it is a special character. An escape sequence is considered as a single character.

**String escape sequences are as follows:**

\n – Newline character  
\b – Backspace  
\\ – Backslash  
\’ – Single quote  
\’’ – Double Quote

**Q #27) What are Regular expressions? Search a string using regular expressions?**

**Ans: Regular expression** is a template to match a set of input. The pattern can consist of operators, constructs or character literals. Regex is used for string parsing and replacing the character string.

**For Example:**

\* matches the preceding character zero or more times. So, a\*b regex is equivalent to b, ab, aab, aaab and so on.

Searching a string using Regex

static void Main(string[] args)

{

string[] languages = { "C#", "Python", "Java" };

foreach(string s in languages)

{

if(System.Text.RegularExpressions.Regex.IsMatch(s,"Python"))

{

Console.WriteLine("Match found");

}

}

}

The above example searches for “Python” against the set of inputs from the languages array. It uses Regex.IsMatch which returns true in case if the pattern is found in the input. The pattern can be any regular expression representing the input that we want to match.

**Q #28) What are the basic String Operations? Explain.**

**Ans: Some of the basic string operations are:**

* **Concatenate** –Two strings can be concatenated either by using System.String.Concat or by using + operator.
* **Modify** – Replace(a,b) is used to replace a string with another string. Trim() is used to trim the string at the end or at the beginning.
* **Compare** – System.StringComparison() is used to compare two strings, either case-sensitive comparison or not case sensitive. Mainly takes two parameters, original string, and string to be compared with.
* **Search** – StartWith, EndsWith methods are used to search a particular string.

**Q #29) What is Parsing? How to Parse a Date Time String?**

**Ans: Parsing** is converting a string into another data type.

**For Example:**

*string text = “500”;*

*int num = int.Parse(text);*

500 is an integer. So, Parse method converts the string 500 into its own base type, i.e int.

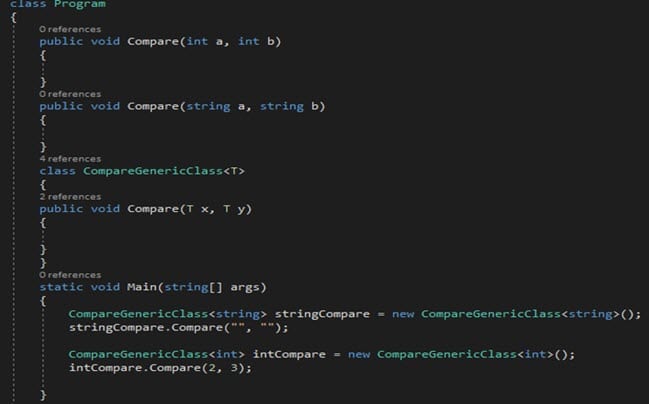
**Follow the same method to convert a DateTime string.**  
string dateTime = “Jan 1, 2018”;  
DateTime parsedValue = DateTime.Parse(dateTime);

Advanced Concepts

**Q #38) What is a Generic Class?**

**Ans:** Generics or Generic class is used to create classes or objects which do not have any specific data type. The data type can be assigned during runtime, i.e when it is used in the program.

**For Example:**

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2018/02/38a.jpg)

So, from the above code, we see 2 compare methods initially, to compare string and int.

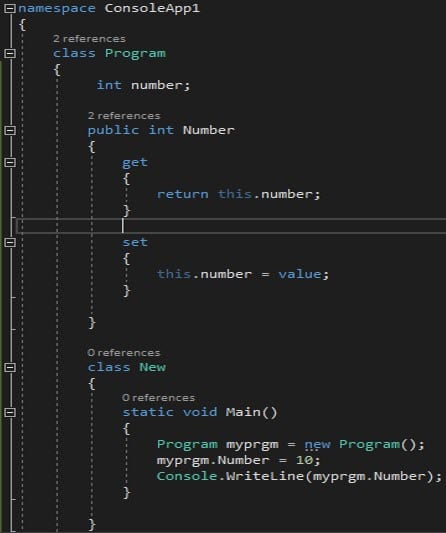
In case of other data type parameter comparisons, instead of creating many overloaded methods, we can create a generic class and pass a substitute data type, i.e T. So, T acts as a datatype until it is used specifically in the Main() method.

**Q #39) Explain Get and Set Accessor properties?**

**Ans: Get and Set** are called Accessors. These are made use by Properties. A property provides a mechanism to read, write the value of a private field. For accessing that private field, these accessors are used.

Get Property is used to return the value of a property  
Set Property accessor is used to set the value.

**The usage of get and set is as below:**

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2018/02/39a.jpg)

**Q #48) What is Serialization?**

**Ans: Serialization** is a process of converting a code to its binary format. Once it is converted to bytes, it can be easily stored and written to a disk or any such storage devices. Serializations are mainly useful when we do not want to lose the original form of the code and it can be retrieved anytime in the future.

Any class which is marked with the attribute [Serializable] will be converted to its binary form.

The reverse process of getting the c# code back from the binary form is called **Deserialization**.

To Serialize an object we need the object to be serialized, a stream which can contain the serialized object and namespace System.Runtime.Serialization can contain classes for serialization.