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

- A Class is an encapsulation of properties and methods
- An Object in an instance of a Class.

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.
- 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.
- Polymorphism – It is achieved by having multiple methods with the same name but different implementations.

Q #3) What is Managed and Unmanaged code?

- 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.**

Q #4) What is an Interface?

An Interface is a class with no implementation.

The only thing that it contains is the declaration of

- Methods
- Properties
- 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.

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.

Abstract class – It is a class whose object cannot be instantiated. The class can only be inherited. It should contain at least one method.

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

Supports Inheritance, Does not support Inheritance

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.

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#.

Q #10) What is "using" statement in C#?

Ans: "Using" Keyword denotes that the particular namespace is being used by the program.

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.

- **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 #15) What is StreamReader/StreamWriter class?

Ans: StreamReader and StreamWriter are classes of namespace System.IO. They are used when we want to read or write charact90, Reader-based data, respectively.

- Some of the members of StreamReader are: Close(), Read(), Readline().
- Members of StreamWriter are: Close(), Write(), Writeline().

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

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.

```
~Car(){
Console.writeline("….");
}
```

Q #17) What is an Abstract Class?

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.

view sourceprint?

```
abstract class AB1
{
Public void Add();
}
Class childClass : AB1
{
childClass cs = new childClass ();
int Sum = cs.Add();
}
```

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?

- 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?

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.

Q #22) What is a Jagged Array?

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

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 #30) What is a Delegate? Explain.

Ans: A **Delegate** is a variable **that holds the reference to a method.** All Delegates are derived from System.Delegate namespace.

public delegate void AddNumbers(int n);

After the declaration of a delegate, the object must be created of the delegate using the new keyword.

AddNumbers an1 = new AddNumbers(number);

Q #31) What are Events?

Ans: Events are user actions that generate notifications to the application to which it must respond. Delegates are used to declare Events.

Public delegate void PrintNumbers();

Event PrintNumbers myEvent;

Q #32) How to use Delegates with Events?

Ans: Delegates are used to raise events and handle them. Always a delegate needs to be declared first and then the Events are declared.

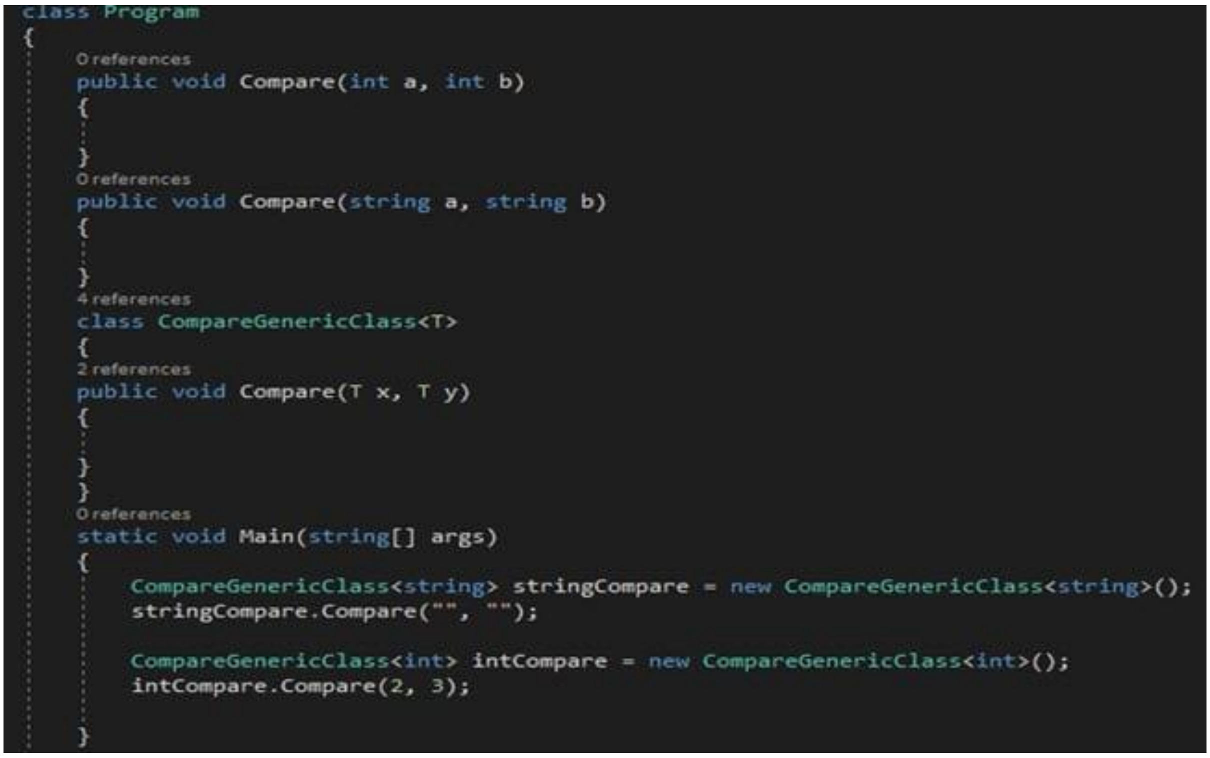
Q #33) What are the different types of Delegates?

Ans: The Different types of Delegates are:

- **Single Delegate** – A delegate which can call a single method.
- **Multicast Delegate** – A delegate which can call multiple methods. + and – operators are used to subscribe and unsubscribe respectively.
- **Generic Delegate** – It does not require an instance of delegate to be defined. It is of three types, Action, Funcs and Predicate.

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.



```
class Program
{
    0 references
    public void Compare(int a, int b)
    {

    }
    0 references
    public void Compare(string a, string b)
    {

    }
    4 references
    class CompareGenericClass<T>
    {
    2 references
    public void Compare(T x, T y)
    {

    }
    }
    0 references
    static void Main(string[] args)
    {
        CompareGenericClass<string> stringCompare = new CompareGenericClass<string>();
        stringCompare.Compare("", "");

        CompareGenericClass<int> intCompare = new CompareGenericClass<int>();
        intCompare.Compare(2, 3);

    }
```

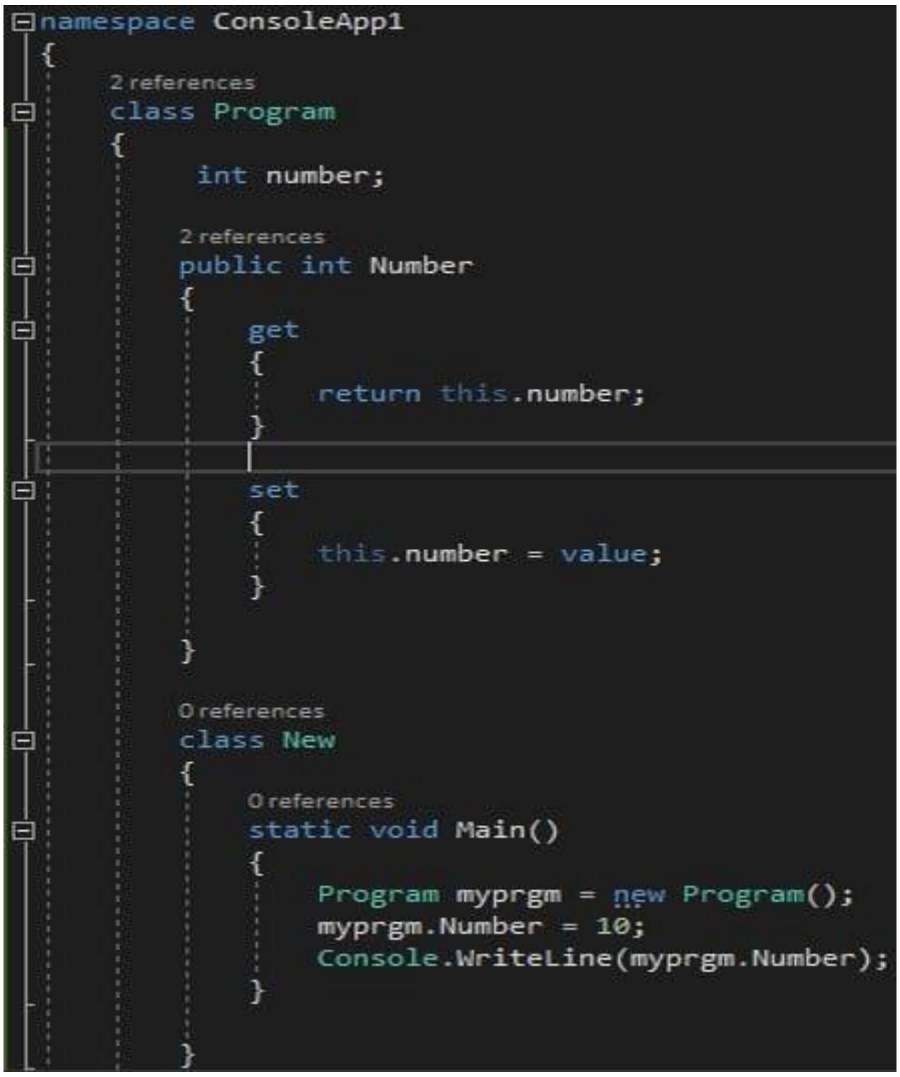
So, from the above code, we see 2 compare methods initially, to compare string and int.

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?

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.



```
namespace ConsoleApp1
{
    2 references
    class Program
    {
        int number;

        2 references
        public int Number
        {
            get
            {
                return this.number;
            }

            set
            {
                this.number = value;
            }

        }

        0 references
        class New
        {
            0 references
            static void Main()
            {
                Program myprgm = new Program();
                myprgm.Number = 10;
                Console.WriteLine(myprgm.Number);
            }

        }
```

Q #40) What is a Thread? What is Multithreading?

A **Thread** is a set of instructions that can be executed, which will enable our program to perform concurrent processing. Concurrent processing helps us do more than one operation at a time. By default, C# has only one thread. But the other threads can be created to execute the code in parallel with the original thread.

Q #41) Name some properties of Thread Class.

Ans: Few Properties of thread class are:

- **IsAlive** – contains value True when a thread is Active.
- **Name** – Can return the name of the thread. Also, can set a name for the thread.
- **Priority** – returns the prioritized value of the task set by the operating system.
- **IsBackground** – gets or sets a value which indicates whether a thread should be a background process or foreground.
- **ThreadState**– describes the thread state.

Q #42) What are the different states of a Thread?

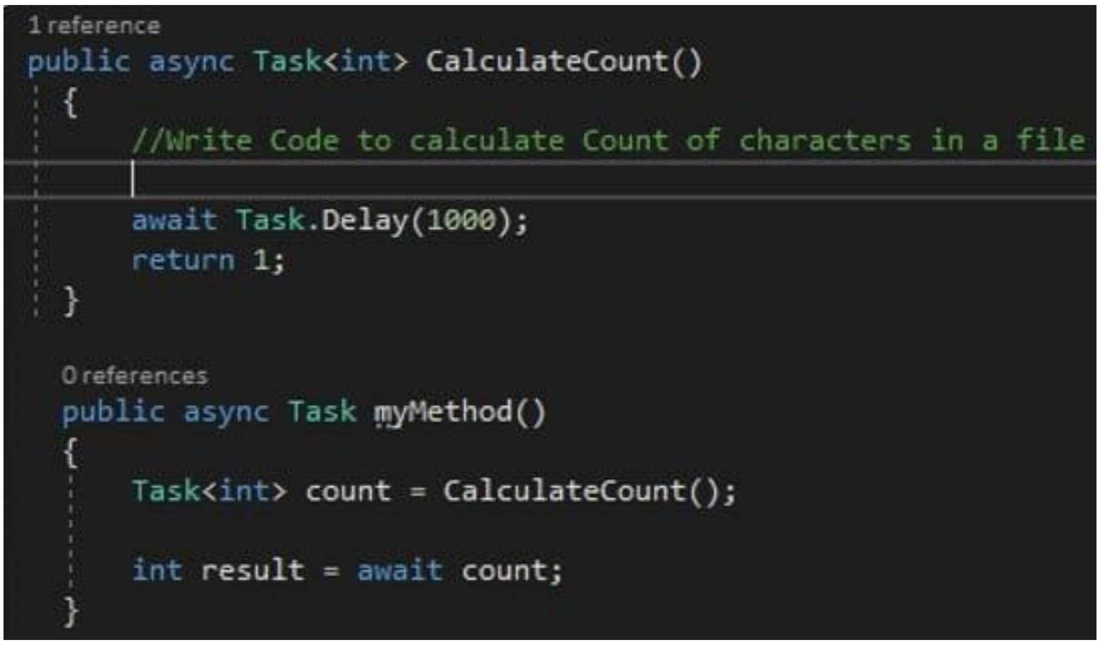
Different states of a thread are:

- **Unstarted** – Thread is created.
- **Running** – Thread starts execution.
- **WaitSleepJoin** – Thread calls sleep, calls wait on another object and calls join on another thread.
- **Suspended** – Thread has been suspended.
- **Aborted** – Thread is dead but not changed to state stopped.
- **Stopped** – Thread has stopped.

Q #43) What are Async and Await?

Async and Await keywords are used to create asynchronous methods in C.

Asynchronous programming means that the process runs independently of main or other processes.



```
1 reference
public async Task<int> CalculateCount()
{
    //Write Code to calculate Count of characters in a file

    await Task.Delay(1000);
    return 1;
}

 0 references
 public async Task myMethod()
 {
    Task<int> count = CalculateCount();

    int result = await count;
 }
```

- Async keyword is used for the method declaration.
- The count is of a task of type int which calls the method CalculateCount().
- Calculatecount() starts execution and calculates something.
- Independent work is done on my thread and then await count statement is reached.
- If the Calculatecount is not finished, myMethod will return to its calling method, thus the main thread doesn't get blocked.
- If the Calculatecount is already finished, then we have the result available when the control reaches await count. So the next step will continue in the same thread. However, it is not the situation in the above case where Delay of 1 second is involved.

Q #44) What is a Deadlock?

A **Deadlock** is a situation where a process is not able to complete its execution because two or more processes are waiting for each other to finish. This usually occurs in multi-threading.

Q #47) What is Thread Pooling?

A **Thread pool** is a collection of threads. **These threads can be used to perform tasks without disturbing the primary thread.** Once the thread completes the task, the thread returns to the pool.

System.Threading.ThreadPool namespace has classes which manage the threads in the pool and its operations.

```
System.Threading.ThreadPool.QueueUserWorkItem(new
System.Threading.WaitCallback(SomeTask));
```

The above line queues a task. SomeTask methods should have a parameter of type Object.