**c# 6.0**

-> Getter Auto property

public int x { get; }

-> string interpolation

return string.format("{0},{1}",x,y);

return $"({x},{y})";

-> Null Condition Operator

abc[""]?.Value == 0

-> Exception Filters

try{}

catch(IOException e) when (e.IsSevere)

**Anonymous Types :**

Used to select limited data (properties) from collection of large objects.

Allows us to specify properties that we want to work with without creating new class

var result = from p in Person

Where p.name.startswith("a")

select new { fname = p.Firstname, Lname = p.LastName };

var obj = new{ a=1;b="Ankit"};

Anonymous Types provides a way to encapsulate read only properties in to a single object

without defining the type of that object

mostly used to store LINQ resultsset

Its internal sealed class

We cant write extention methods

We cant be use it as parameter or return type

**Static and Singletone**

Static Class:

You cannot create the instance of static class.

Loaded automatically by the .NET Framework common language runtime (CLR) when the program or namespace containing the class is loaded.

Static Class cannot have constructor.

We cannot pass the static class to method.

We cannot inherit Static class to another Static class in C#.

Singleton:

You can create one instance of the object and reuse it.

Singleton instance is created for the first time when the user requested. 

Singleton class can have constructor.

You can create the object of singleton class and pass it to method.

Singleton class does not say any restriction of Inheritance.

We can dispose the objects of a singleton class but not of static class.

Singleton objects are stored in Heap, but static objects are stored in stack.

We can clone (if the designer did not disallow it) the singleton object, but we can not clone the static class object .

Singleton classes follow the OOP (object oriented principles), static classes do not.

We can implement an interface with a Singleton class, but a class's static methods (or e.g. a C# static class) cannot.

**Thread Safe Techniques**

**Lock/Monitor/ Mutex/Semaphore/**

Lock keyword is just a shot-form of Monitor. So usage of monitor class or lock is one and same.

static readonly object \_object = new object();

lock(\_object)

{

Thread safe code

}

Monitor.Enter(\_object);

//thread safe code

Monitor.Exit(\_object);

Locks/Monitors ensures the thread safety which are in process that is threads which are generated by an application

Mutex ensures the thread safety which are out process that is threads which coming from outside of an application