Different types of class in C# programming

What are the types of class?

There are four different types of class.

1. Abstract class
2. Partial Class
3. Sealed class
4. Static class

What is abstract class?

A class with abstract modifier indicate that class is abstract class. An abstract class cannot be instantiated. The purpose of an abstract class is to provide a common definition of a base class that multiple derived classes can share.

Characteristic of Abstract class.

1. An abstract class cannot be instantiated.
2. An abstract class may contain abstract methods and accessors.
3. An abstract class cannot be sealed. The sealed modifier prevents a class from being inherited and the abstract modifier requires a class to be inherited.
4. A non-abstract class derived from an abstract class must include actual implementations of all inherited abstract methods and accessors.

```
using System;
namespace AbstractClass_demo
{
    public abstract class Customer
    {
        private string _firstName;
        private string _lastName;

        public string FirstName
        {
            get
            {
                return _firstName;
            }
            set
            {
                _firstName = value;
            }
        }

        public string LastName
        {
            get
            {
                return _lastName;
```

```
            }
            set
            {
                _lastName = value;
            }
        }

        public abstract void FullName();
    }
}

using AbstractClass_Demo;
using System;

namespace AbstractClass_Demo
{
    class Program: Customer
    {
        static void Main(string[] args)
        {
            Program p = new Program();
            p.FirstName = "Farhan";
            p.LastName = "Ahmed";
            p.FullName();
            Console.ReadLine();

        }
        public override void FullName()
        {
            Console.WriteLine("Full Name:"+FirstName + " " +LastName);
        }
    }
}
```

What is partial class?

The partial keyword indicates that other parts of the class, struct, or interface can be defined in the namespace. All the parts must use the partial keyword. All the parts must be available at compile time to form the final type. All the parts must have the same accessibility, such as public, private, and so on.

Characteristic of Partial class.

1. All the partial class definitions must be in the same assembly and namespace.
2. All the parts must have the same accessibility like public or private, etc.
3. If any part is declared abstract, sealed or base type then the whole class is declared of the same type.
4. Different parts can have different base types and so the final class will inherit all the base types.
5. The Partial modifier can only appear immediately before the keywords class, struct, or interface.
6. Nested partial types are allowed.

```
using System;
namespace PartialClass_Demo
{
    public partial class PartialClass
    {
        private string _firstName;
        private string _lastName;
        public string FirstName
        {
            get
            {
                return _firstName;
            }

            set
            {
                _firstName = value;
            }
        }
        public string LastName
        {
            get
            {
                return _lastName;
            }

            set
            {
                _lastName = value;
            }
        }
    }
}
```

```
using System;
namespace PartialClass_Demo
{
    public partial class PartialClass
    {
        public void FullName()
        {
            Console.WriteLine("Full Name:"+FirstName+" "+LastName );
        }
    }
}
```

```
using System;
namespace PartialClass_Demo
{
    class Program
    {
        static void Main(string[] args)
        {
            PartialClass partial = new PartialClass();
            partial.FirstName = "Farhan";
            partial.LastName = "Ahmed";
            partial.FullName();

            Console.ReadLine();
        }
    }
}
```

What is sealed class?

A class with sealed keyword indicate that class is sealed to prevent inheritance. Sealed class cannot inheritance.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SealedClass_Demo
{
    public sealed class Employee
    {
        string firstName;
        string lastName;
    }
    class Program:Employee
    {
        static void Main(string[] args)
        {

        }
    }
}
```

What is static class?

A class with static keyword and contains only static members defined as static class. A static class cannot be instantiated.

Characteristic of static class.

1. Static class cannot instantiated using new keyword.
2. Static items can only access other static items. For example, a static class can only contain static members, e.g. variable, methods etc.
3. A static method can only contain static variables and can only access other static items.
4. Static items share the resources between multiple users.
5. Static cannot be used with indexers, destructors or types other than classes.
6. A static constructor in a non-static class runs only once when the class is instantiated for the first time.
7. A static constructor in a static class runs only once when any of its static members accessed for the first time.
8. Static members are allocated in high frequency heap area of the memory.

```
using System;

namespace StaticClass_Demo
{
    public static class HeightConvertor
    {
        public static double InchsToCentimeters(string HeightInInchs)
        {
            double inchs = Double.Parse(HeightInInchs);
            double Centimeters = (inchs* 2.54);
            return Centimeters;
        }

        public static double CentimetesToInchs(string HeightInCentimeters)
        {
            double centimeters = Double.Parse(HeightInCentimeters);
            double Inchs = (centimeters / 2.54);
            return Inchs;
        }
    }
}
```

```
using System;
namespace StaticClass_Demo
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Please select the convertor direction");
            Console.WriteLine("1. From Inchs to Centimeters.");
            Console.WriteLine("2. From Centimeters to Inchs.");

            string selection = Console.ReadLine();
            double C, I = 0;

            switch (selection)
            {
                case "1":
                    Console.Write("Please enter the height in inchs: ");
                    C = HeightConvertor.InchsToCentimeters(Console.ReadLine());
                    Console.WriteLine("Hieght in centimeters: {0:F2}", C);
                    break;

                case "2":
                    Console.Write("Please enter the Height in centimeters: ");
                    I = HeightConvertor.CentimetesToInchs(Console.ReadLine());
                    Console.WriteLine("Height in Inchs: {0:F2}", I);
                    break;

                default:
                    Console.WriteLine("Please select a convertor.");
                    break;
            }

            Console.WriteLine("Press any key to exit.");
            Console.ReadLine();
        }
    }
}
```