```
namespace CollectionsDemo
{
    internal class Person
    {
        public Person()
        {
        }
    }
}
```

[970718](#)/[CollectionsDemo](#)/**Program.cs**

```
using System;
using System.Collections;
namespace CollectionsDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            ArrayList al = new ArrayList();

            al.Add(123);
            al.Add(456);
            al.Add("Test");
            al.Add(DateTime.Now);
            al.Add(new Person());
            al.Add(new Random());

            //al.Sort();

            //var result = al[0] is Person;
            //var p = al[0] as Person;
            //var result = p == null ? false : true;
            //var p = (Person)al[0];

            al.Remove(456);
            al.RemoveAt(0);
            Console.WriteLine(al.IndexOf("Test"));
            al.Insert(1, "Inserted member");
            foreach (var item in al)
            {
                Console.WriteLine(item);
            }


            Console.WriteLine(al.Contains("Test"));
            al.Clear();
            Console.WriteLine(al.Count);

            Console.ReadKey();
```

970718/GenericClassDemo/Person.cs

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

namespace GenericClassDemo
{
    class Person
    {
    }
}
```

970718/GenericClassDemo/Program.cs

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

namespace GenericClassDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            Repository<Person> repository = new Repository<Person>();
            repository.Insert(new Person());
            var people = repository.GetAllRecords();
        }
    }
}
```

970718/GenericClassDemo/Repository.cs

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

namespace GenericClassDemo
{
    class Repository<K>
    {
        public K[] GetAllRecords()
        {
            return null;
        }
        public void Insert(K record)
        {
```

970718/ExplicitInterfaceDemo/IFirstInterface.cs

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

namespace ExplicitInterfaceDemo
{
    interface IFirstInterface
    {
        void MyMethod();
        void FirstMethod();
    }
}
```

970718/ExplicitInterfaceDemo/ISecondInterface.cs

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

namespace ExplicitInterfaceDemo
{
    interface ISecondInterface
    {
        void MyMethod();
        void SecondMethod();
    }
}
```

970718/ExplicitInterfaceDemo/Program.cs

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

namespace ExplicitInterfaceDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            MyClass mc = new MyClass();
            //((IFirstInterface)mc)
            IFirstInterface fi = new MyClass();
            ISecondInterface si = new MyClass();


        }
```

970718/ExplicitInterfaceDemo/MyClass.cs

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

namespace ExplicitInterfaceDemo
{
    class MyClass : IFirstInterface, ISecondInterface
    {
        void IFirstInterface.FirstMethod()
        {
            throw new NotImplementedException();
        }

        void IFirstInterface.MyMethod()
        {
            throw new NotImplementedException();
        }

        void ISecondInterface.MyMethod()
        {
            throw new NotImplementedException();
        }

        void ISecondInterface.SecondMethod()
        {
            throw new NotImplementedException();
        }
    }
}
```

70718/GenericCollectionsDemo/Person.cs

```
namespace GenericCollectionsDemo
{
    internal class Person
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Family { get; set; }
    }
}
```

970718/GenericCollectionsDemo/Program.cs

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

namespace GenericCollectionsDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            //List<string> names = new List<string>();
            //List<int> scores = new List<int>();

            //List<Person> people = new List<Person>();

            //Dictionary<string, Person> people = new Dictionary<string, Person>();
            //people.Add("STD01", new Person() { Id = 1, Name = "John", Family =
"Doe" });
            //people.Add("STD02", new Person() { Id = 2, Name = "David", Family =
"Smith" });
            //people.Add("STD03", new Person() { Id = 3, Name = "Ross", Family =
"Geller" });

            //foreach (KeyValuePair<string,Person> item in people)
            //{
            //    Console.WriteLine($"{item.Key} {item.Value.Name}");
            //}

            //Stack<int> s = new Stack<int>();
            //s.Push(1);
            //s.Push(2);
            //s.Push(3);

            //Console.WriteLine(s.Pop());
            //Console.WriteLine(s.Pop());
            //Console.WriteLine(s.Pop());

            Queue<int> q = new Queue<int>();
            q.Enqueue(1);
            q.Enqueue(2);
            q.Enqueue(3);

            Console.WriteLine(q.Dequeue());
            Console.WriteLine(q.Dequeue());
            Console.WriteLine(q.Dequeue());



            Console.ReadKey();
        }
```

970718/GenericInterfaceDemo/Exam.cs

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

namespace GenericInterfaceDemo
{
    class Exam:IEnumerable<Question>
    {
        public Question[] Questions { get; set; }

        public IEnumerator<Question> GetEnumerator()
        {
            throw new NotImplementedException();
        }

        IEnumerator IEnumerable.GetEnumerator()
        {
            throw new NotImplementedException();
        }
    }

    class ExamEnumerator : IEnumerator<Question>
    {
        public Question Current => throw new NotImplementedException();

        object IEnumerator.Current => throw new NotImplementedException();

        public void Dispose()
        {
            throw new NotImplementedException();
        }

        public bool MoveNext()
        {
            throw new NotImplementedException();
        }

        public void Reset()
        {
            throw new NotImplementedException();
        }
    }
}
```

970718/GenericInterfaceDemo/Program.cs

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

namespace GenericInterfaceDemo
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

970718/GenericInterfaceDemo/Question.cs

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

namespace GenericInterfaceDemo
{
    class Question : IComparable<Question>
    {
        public int CompareTo(Question other)
        {
            throw new NotImplementedException();
        }
    }
}
```

[970718](#)/[IComparableDemo](#)/**Student.cs**

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

namespace IComparableDemo
{
    class Student:IComparable
    {
        public string Name { get; set; }
        public string Family { get; set; }
        public double Average { get; set; }

        public int CompareTo(object obj)
        {
            var other = obj as Student;
            if (other != null)//Student
            {
                if (this.Family != other.Family)
                    return this.Family.CompareTo(other.Family);

                return this.Name.CompareTo(other.Name);
            }
            return -1;
        }

        public override string ToString()
        {
            return $"{Name} {Family} [{Average}]";
        }
    }

    class StudentAverageComparer : IComparer
    {
        public int Compare(object x, object y)
        {
            return (((Student)x).Average).CompareTo(((Student)y).Average) * -1;
        }
    }
}
```

970718/IComparableDemo/Program.cs

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

namespace IComparableDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            Student[] students = new Student[]
                {
                    new Student() { Name = "Ross", Family = "Geller", Average = 80 },
                    new Student() { Name = "Chandler", Family = "Bing", Average = 70
},

                    new Student() { Name = "Monica", Family = "Geller", Average = 90
}
                };

            Array.Sort(students, new StudentAverageComparer());

            //IComparer comparer = new StudentAverageComparer();


            foreach (Student student in students)
            {
                Console.WriteLine(student);
            }

            Console.ReadKey();
        }
    }
}
```

970718/IOInfoClassesDemo/Program.cs

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

namespace IOInfoClassesDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            //DriveInfo[] drives = DriveInfo.GetDrives();
            //for (int i = 0; i < drives.Length; i++)
            //{
            //    if(drives[i].IsReady)
            //
Console.WriteLine($"{i+1}.{drives[i].Name}({drives[i].VolumeLabel})-
[{drives[i].DriveType}]\t{drives[i].AvailableFreeSpace/1024/1024/1024} GB /
{drives[i].TotalSize/1024/1024/1024} GB");
            //    else
            //        Console.WriteLine($"{i+1}.{drives[i].Name}-
[{drives[i].DriveType}]");
            //}

            //DirectoryInfo di = new DirectoryInfo(@"D:\");
            //DirectoryInfo[] subDirectories = di.GetDirectories();
            //foreach (var directory in subDirectories)
            //{
            //    Console.WriteLine($"{directory.Name}\t{directory.FullName}");
            //}

            //FileInfo[] fi = di.GetFiles();
            //foreach (var item in fi)
            //{
            //    Console.WriteLine($"{item.Name}");
            //}

            //DirectoryInfo di = new
DirectoryInfo(@"D:\Development\Courses\CourseCodes\Pack-9705");
            //Console.WriteLine(di.Parent);

            //DirectoryInfo di = new DirectoryInfo(@"D:\");
            //Console.WriteLine(di.Parent);

            Process.Start(@"D:\EBooks\IllustratedCSharp7,5thEdition@farhangv_com.pdf"
);

            Console.ReadKey();
        }
```

970718/RegexDemo/Program.cs

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

namespace RegexDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            while (true)
            {
                Console.Clear();
                Console.Write("Pattern:");
                var pattern = Console.ReadLine();
                if (pattern == "exit")
                    break;
                Regex re = new Regex(pattern);
                while (true)
                {
                    Console.Write("Text:");
                    var text = Console.ReadLine();
                    if (text == "exit")
                        break;
                    var isMatch = re.IsMatch(text);
                    if (isMatch)
                    {
                        Console.ForegroundColor = ConsoleColor.Green;
                        Console.WriteLine("Match!");
                    }
                    else
                    {
                        Console.ForegroundColor = ConsoleColor.Red;
                        Console.WriteLine("Not Match!");
                    }
                    Console.ForegroundColor = ConsoleColor.White;
                }
            }

        }
    }
}
```

970718/StreamReaderWriterDemo/Program.cs

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

namespace StreamReaderWriterDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            //using (StreamWriter sw = new StreamWriter(@"mydata.txt", true))
            //{
            //    //sw.AutoFlush = true;
            //    while (true)
            //    {
            //        var input = Console.ReadLine();
            //        if (input == "exit")
            //            break;
            //        sw.WriteLine(input);
            //        //sw.Flush();
            //    }
            //}//sw.Close();

            List<string> names = new List<string>();
            using (StreamReader sr = new StreamReader(@"mydata.txt"))
            {
                //Console.WriteLine((char)sr.Read());
                //Console.WriteLine((char)sr.Read());
                //Console.WriteLine((char)sr.Read());
                //Console.WriteLine((char)sr.Read());
                //Console.WriteLine(Convert.ToChar(sr.Read()));

                //while (sr.Peek() != -1)
                //{
                //    Console.Write((char)sr.Read());
                //}

                while (sr.Peek() != -1)
                {
                    //Console.WriteLine(sr.ReadLine());
                    names.Add(sr.ReadLine());
                }
                //Console.WriteLine(sr.ReadToEnd());

            }
            Console.ReadKey();
        }
        }
}
```