Container Object

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

namespace X04_MyContainer_object
{
    public class MyContainer
    {
        private object[] _theObjects;
        private int _n;

        public MyContainer()
        {
            _theObjects = new object[2];
            _n = 0;
        }

        public void Add(object o)
        {
            // If necessary, grow the array
            if (_n == _theObjects.Length)
            {
                object[] oldArray = _theObjects;
                _theObjects = new object[2 * oldArray.Length];
                Array.Copy(oldArray, _theObjects, _n);
            }

            _theObjects[_n] = o;
            _n++;
        }

        public object GetAt(int i)
        {
            return _theObjects[i];
        }

        public int Count
        {
            get { return _n; }
        }
    }


    class Program
    {
        static void Main(string[] args)
        {
            MyContainer container = new MyContainer();

            container.Add(3);
            container.Add(2);
            container.Add(8);
            container.Add(8);
            container.Add(4);

            for (int i = 0; i < container.Count; i++)
            {
                Console.WriteLine($"Element at {i}: {container.GetAt(i)}");
            }
            Console.ReadKey();
        }
    }
}
```

Container Generic

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

namespace X04_MyContainer_generic
{
   public class MyContainer<T> : IEnumerable<T>
    {
        private T[] _theObjects;
        private int _n;

        public MyContainer()
        {
            _theObjects = new T[2];
            _n = 0;
        }

        public void Add(T o)
        {
            // If necessary, grow the array
            if (_n == _theObjects.Length)
            {
                T[] oldArray = _theObjects;
                _theObjects = new T[2 * oldArray.Length];
                Array.Copy(oldArray, _theObjects, _n);
            }

            _theObjects[_n] = o;
            _n++;
        }

        public T GetAt(int i)
        {
            return _theObjects[i];
        }

        public int Count
        {
            get { return _n; }
        }

        public IEnumerator<T> GetEnumerator()
        {
            for (int i = 0; i < _n; i++)
            {
                yield return _theObjects[i];
            }
        }

        IEnumerator IEnumerable.GetEnumerator()
        {
            return GetEnumerator();
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            MyContainer<int> container = new MyContainer<int>();

            container.Add(3);
            container.Add(2);
            container.Add(8);
            container.Add(8);
            container.Add(4);

            for (int i = 0; i < container.Count; i++)
            {
                Console.WriteLine($"Element at {i}: {container.GetAt(i)}");
            }
            Console.ReadKey();
        }
    }
}
```