```
/*--------------------------------------------------------------------------------------------------
 * Name:        Dan Cassidy
 * Date:        2015-06-09
 * Assignment:  cView-P2
 * Source File: CViewDataSet.cs
 * Course:      CSCI-C 490, C# Programming, MoWe 08:00
 * Purpose:     Encapsulates a List-based collection of CViewData objects and contains related
 *              methods and properties.
--------------------------------------------------------------------------------------------------*/

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

namespace CView
{
    class CViewDataSet
    {
        //Basic field of the class.
        private List<CViewData> dataSet = new List<CViewData>();

        //Enable read-only access to the Count property.
        public int Count
        {
            get
            {
                return dataSet.Count;
            }
        }

        //Enable read-only access to the Header property. Uses the header from the CViewData class
        //so if needs to be changed, it only needs to be changed in one place.
        public string Header
        {
            get
            {
                return CViewData.HEADER;
            }
        }

        /*------------------------------------------------------------------------------------------
         * Method:  this[]
         * Purpose: Access the objects in this dataset via index number.
         * Input:   int objectNum, the index of the object that will be accessed.
         * Output:  CViewData object of the referenced object at the index.
        ------------------------------------------------------------------------------------------*/
        public CViewData this[int objectNum]
        {
            get
            {
                //Try to simply return the object at index objectNum.
                try
                {
                    return dataSet[objectNum];
                }
                catch (ArgumentOutOfRangeException)
                {
                    //If this exception is caught, let the user know and return a null.
                    Console.WriteLine("Index [{0}] is out of range.", objectNum);
                    return null;
                }
            }
            set
            {
                //Try to set the object at index objectNum.
                try
                {
                    dataSet[objectNum] = value;
                }
                catch (ArgumentOutOfRangeException)
                {
                    //If this exception is caught, do nothing further and let the user know.
                    Console.WriteLine("Index [{0}] is out of range.", objectNum);
                }
            }
        }

        /*------------------------------------------------------------------------------------------
         * Method:  Add
         * Purpose: Add a data object to the dataset.
         * Input:   CViewData toAdd, this is the object that will get added to the dataset.
         * Output:  Nothing.
        ------------------------------------------------------------------------------------------*/
        public void Add(CViewData toAdd)
        {
            //Add object using List Add method.
            dataSet.Add(toAdd);
        }

        /*------------------------------------------------------------------------------------------
         * Method:  Delete
         * Purpose: Delete an object at the given index from the dataset.
         * Input:   int indexToRemove, the index of the object to be removed from the dataset.
         * Output:  Nothing.
        ------------------------------------------------------------------------------------------*/
        public void Delete(int indexToRemove)
        {
            //Delete object at specified index by using List RemoveAt method.
            dataSet.RemoveAt(indexToRemove);
        }

        /*------------------------------------------------------------------------------------------
         * Method:  Search
         * Purpose: Search for a given string in this dataset.
         * Input:   string toSearchFor, this is the string that will be searched for.
         * Input:   CViewData.Fields searchField, this is the field that will be searched.
         * Output:  CViewDataSet object, containing all (if any) objects found.
        ------------------------------------------------------------------------------------------*/
        public CViewDataSet Search(string toSearchFor, CViewData.Fields searchField)
        {
            //Shortened form of StringComparison.OrdinalIgnoreCase for code prettiness.
            var ignoreCase = StringComparison.OrdinalIgnoreCase;

            //Use LINQ to search the objects with case insensitivity. Basic case insitivity code
            //idea from Stack Overflow. http://stackoverflow.com/a/444818
            var foundData =
                from data in dataSet
                where
                    //Search Name property.
                    (searchField == CViewData.Fields.Name &&
                     data.Name.IndexOf(toSearchFor, ignoreCase) >= 0) ||
                    //Search FacilityType property.
                    (searchField == CViewData.Fields.FacilityType &&
                     data.FacilityType.IndexOf(toSearchFor, ignoreCase) >= 0) ||
                    //Search Address property.
                    (searchField == CViewData.Fields.Address &&
                     data.Address.IndexOf(toSearchFor, ignoreCase) >= 0) ||
                    //Search City property.
                    (searchField == CViewData.Fields.City &&
                     data.City.IndexOf(toSearchFor, ignoreCase) >= 0) ||
                    //Search PhoneNumber propery.
                    (searchField == CViewData.Fields.PhoneNumber &&
                     data.PhoneNumber.IndexOf(toSearchFor, ignoreCase) >= 0)
                select data;

            //Return a new dataset containing the found objects.
            return new CViewDataSet() { dataSet = foundData.ToList() };
        }

        /*------------------------------------------------------------------------------------------
         * Method:  SortByName
         * Purpose: Sort the dataset by the Name property of the objects.
         * Input:   Nothing.
         * Output:  Nothing.
        ------------------------------------------------------------------------------------------*/
        public void SortByName()
        {
            //Idea from Stack Overflow: http://stackoverflow.com/a/3309230
            //Yay lambda expressions!
            dataSet = dataSet.OrderBy(data => data.Name).ToList();
        }

        /*------------------------------------------------------------------------------------------
         * Method:  ToString
         * Purpose: Override of the ToString() method. Formats the return value so it looks pretty.
         * Input:   Nothing.
         * Output:  String object containing serialized collection data.
        ------------------------------------------------------------------------------------------*/
        public override string ToString()
        {
            //Declare the string.
            string toReturn = "";

            //Build the string.
            foreach (var item in dataSet)
                toReturn += item.ToString() + "\n";

            //Return the string.
            return toReturn;
        }
    }
}
```