```
/*-----------------------------------------------------------------------------------------------
 * Name:        Dan Cassidy
 * Date:        2015-06-02
 * Assignment:  cView-P1
 * Source File: CViewDataInteractive.cs
 * Class:       CSCI-C 490, C# Programming, MoWe 08:00
 * Purpose:     Provides interactive management of a CViewDataSet object.
-----------------------------------------------------------------------------------------------*/

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

namespace cView_P1_DanCassidy
{
    class CViewDataInteractive
    {
        private CViewDataSet data = new CViewDataSet();

        //Helper constants for menu validation.
        private const mainMenu MAINMENU_MIN = mainMenu.ADD;
        private const mainMenu MAINMENU_MAX = mainMenu.EXIT;
        private const modifyMenu MODIFYMENU_MIN = modifyMenu.NAME;
        private const modifyMenu MODIFYMENU_MAX = modifyMenu.BACK;

        //Enum for the main menu. Basic code idea from Stack Overflow.
        //http://stackoverflow.com/a/15752719
        private enum mainMenu
        {
            ADD = 1,
            MODIFY,
            SEARCH,
            DELETE,
            DISPLAY_ALL,
            EXIT
        }

        //Enum for the modify menu. Basic code idea from Stack Overflow.
        //http://stackoverflow.com/a/15752719
        private enum modifyMenu
        {
            NAME = 1,
            ADDRESS,
            CITY,
            STATE,
            ZIP,
            PHONE,
            BACK
        }

        /*---------------------------------------------------------------------------------------
         * Method:  InteractiveManipulation
         * Purpose: Entry point for interactive manipulation of CViewDataSet object.
         * Input:   Nothing.
         * Output:  Nothing.
        ---------------------------------------------------------------------------------------*/
        public void InteractiveManipulation()
        {
            //Loop the main menu until the user decides to exit.
            while (MainMenuAction(MainMenuDisplay()) != mainMenu.EXIT) ;
        }

        /*---------------------------------------------------------------------------------------
         * Method:  MainMenuDisplay
         * Purpose: Display the main menu and get a choice. Must have valid input to return.
         * Input:   Nothing.
         * Output:  mainMenu, representing the choice that was made.
        ------------------------------------------------------------------------------------------*/
        private mainMenu MainMenuDisplay()
        {
            mainMenu menuChoice;
            bool invalid;

            do
            {
                //Display the menu.
                Console.WriteLine("------------------------");
                Console.WriteLine("| Main Interactive Menu |");
                Console.WriteLine("------------------------");
                Console.WriteLine("Please select an option:");
                Console.WriteLine("  1) Add New Item");
                Console.WriteLine("  2) Modify Item");
                Console.WriteLine("  3) Search Items");
                Console.WriteLine("  4) Delete Item");
                Console.WriteLine("  5) Display All Items");
                Console.WriteLine("  6) Exit");
                Console.Write("Choice: ");

                //Get the user's choice.
                string input = Console.ReadLine();

                //Extra line for formatting.
                Console.WriteLine();

                //Validate the user input.
                invalid = !mainMenu.TryParse(input, out menuChoice) ||
                          !MainMenuValidate(menuChoice);
            } while (invalid);

            //Return the user's choice.
            return menuChoice;
        }

        /*------------------------------------------------------------------------------------------
         * Method:  MainMenuValidate
         * Purpose: Validates that the choice by the user is within the limits and is logically
         *          possible.
         * Input:   mainMenu value, contains the user's choice.
         * Output:  bool, representing whether the user's choice was valid or not.
        ------------------------------------------------------------------------------------------*/
        private bool MainMenuValidate(mainMenu value)
        {
            //Check to make sure that the user input is within valid limits.
            if (value < MAINMENU_MIN || value > MAINMENU_MAX)
                return false;

            //If the data set is empty, limit user to adding an entry or exiting.
            if (data.Count == 0 && (value != mainMenu.ADD && value != mainMenu.EXIT))
            {
                Console.WriteLine("No data is present. Please choose a different option.\n");
                return false;
            }

            //Otherwise, input is good.
            return true;
        }

        /*------------------------------------------------------------------------------------------
         * Method:  MainMenuAction
         * Purpose: Acts on the user's choice made at the Main Menu.
         * Input:   mainMenu choice, represents the action specified.
         * Output:  mainMenu, represents the action specified.
        ---------------------------------------------------------------------------------------------*/
        private mainMenu MainMenuAction(mainMenu choice)
        {
            //Decide what to do based on the user's choice.
            switch (choice)
            {
                case mainMenu.ADD:
                    //Add a new item.
                    DataAdd();
                    break;

                case mainMenu.MODIFY:
                    //Modify an existing item.
                    DataModify();
                    break;

                case mainMenu.SEARCH:
                    //Search items.
                    DataSearch();
                    break;

                case mainMenu.DELETE:
                    //Delete an item.
                    DataDelete();
                    break;

                case mainMenu.DISPLAY_ALL:
                    //Display all the items.
                    DataDisplayAll();
                    break;

                case mainMenu.EXIT:
                    //Do nothing, exiting the method.
                default:
                    //Catch-all.
                    break;
            }

            //Return choice so the calling method knows what the choice was and can act accordingly.
            return choice;
        }

        /*---------------------------------------------------------------------------------------------
         * Method:  DataAdd
         * Purpose: Interactively add an item based on the user's input.
         * Input:   Nothing.
         * Output:  Nothing.
        ---------------------------------------------------------------------------------------------*/
        private void DataAdd()
        {
            CViewData tempData = new CViewData();

            //Prompt the user to input information about the new item.
            Console.WriteLine("----------------");
            Console.WriteLine("| Add New Item |");
            Console.WriteLine("----------------");
            Console.Write("Business Name: ");
            tempData.Name = Console.ReadLine();
            Console.Write("Address: ");
            tempData.Address = Console.ReadLine();
            Console.Write("City: ");
            tempData.City = Console.ReadLine();
            Console.Write("State: ");
            tempData.State = Console.ReadLine();
            Console.Write("ZIP Code: ");
            tempData.ZIPCode = Console.ReadLine();
            Console.Write("Phone Number: ");
            tempData.PhoneNumber = Console.ReadLine();


            Console.WriteLine();

            //Add the new item to the main data set.
            data.Add(tempData);

            //Sort the data set.
            data.SortByName();
        }

        /*-----------------------------------------------------------------------------------------
         * Method:  DataModify
         * Purpose: Interactively modifies an object based on the user's input.
         * Input:   Nothing.
         * Output:  Nothing.
        -----------------------------------------------------------------------------------------*/
        private void DataModify()
        {
            //Display the user's choice.
            Console.WriteLine("---------------------------------");
            Console.WriteLine("| Modify Item -- Existing Items |");
            Console.WriteLine("---------------------------------");

            //Display a numbered list of all the objects in the data set.
            DataDisplayAllNumbered();

            //Get the user's choice of which object to delete.
            Console.Write("\nSelect item (0 for none): ");
            int indexToModify = int.Parse(Console.ReadLine()) - 1;

            //Extra line for formatting.
            Console.WriteLine();

            //Validate the user's choice.
            if (indexToModify == -1)
            {
                //The user changed their mind.
                Console.WriteLine("Cancelled.\n");
                return;
            }
            else if (indexToModify < 0 || indexToModify >= data.Count)
            {
                //The user input an invalid object index.
                Console.WriteLine("Invalid item.\n");
                return;
            }

            do
            {
                //Display the chosen object.
                Console.WriteLine("------------------------------");
                Console.WriteLine("| Modify Item -- Chosen Item |");
                Console.WriteLine("------------------------------");
                Console.WriteLine("{0}\n{1}\n", data.Header, data[indexToModify]);

                //Loop while the use has not chosen to go back.
            } while (ModifyMenuAction(ModifyMenuDisplay(), indexToModify) != modifyMenu.BACK);
        }

        /*-----------------------------------------------------------------------------------------
         * Method:  ModifyMenuDisplay
         * Purpose: Display the modify menu and get a choice. Must have valid input to return.
         * Input:   Nothing.
         * Output:  modifyMenu, representing the choice that was made.
        --------------------------------------------------------------------------------------------*/
        private modifyMenu ModifyMenuDisplay()
        {
            modifyMenu menuChoice;
            bool invalid;

            do
            {
                //Display the menu.
                Console.WriteLine("Please select the field you would like to modify:");
                Console.WriteLine("  1) Business Name");
                Console.WriteLine("  2) Street Address");
                Console.WriteLine("  3) City");
                Console.WriteLine("  4) State");
                Console.WriteLine("  5) ZIP Code");
                Console.WriteLine("  6) Phone Number");
                Console.WriteLine("  7) Back");
                Console.Write("Choice: ");

                //Get the user's choice.
                string input = Console.ReadLine();

                //Extra line for formatting.
                Console.WriteLine();

                //Validate the user input.
                invalid = !modifyMenu.TryParse(input, out menuChoice) ||
                          !ModifyMenuValidate(menuChoice);
            } while (invalid);

            //Return the user's choice.
            return menuChoice;
        }

        /*--------------------------------------------------------------------------------------------
         * Method:  ModifyMenuValidate
         * Purpose: Validates that the choice by the user is within the limits and is logically
         *          possible.
         * Input:   mmodifyMenu value, contains the user's choice.
         * Output:  bool, representing whether the user's choice was valid or not.
        --------------------------------------------------------------------------------------------*/
        private bool ModifyMenuValidate(modifyMenu value)
        {
            //Check to make sure that the user input is within valid limits.
            if (value < MODIFYMENU_MIN || value > MODIFYMENU_MAX)
                return false;

            //Otherwise, input is good.
            return true;
        }

        /*--------------------------------------------------------------------------------------------
         * Method:  ModifyMenuAction
         * Purpose: Acts on the user's choice made at the Modify Menu.
         * Input:   modifyMenu choice, represents the action specified.
         * Output:  modifyMenu, represents the action specified.
        --------------------------------------------------------------------------------------------*/
        private modifyMenu ModifyMenuAction(modifyMenu choice, int indexToModify)
        {
            //Decide what to do based on the user's choice.
            switch (choice)
            {
                case modifyMenu.NAME:
                        //Change the name of the item.
                        Console.WriteLine("Current Business Name: {0}", data[indexToModify].Name);
                        Console.Write("New Business Name: ");
                        data[indexToModify].Name = Console.ReadLine();

                        //Extra line for formatting.
                        Console.WriteLine();

                        //Sort the data set after changing the name since name is the primary
                        //sort criteria.
                        data.SortByName();

                        break;

                    case modifyMenu.ADDRESS:
                        //Change the address of the item.
                        Console.WriteLine("Current Address: {0}", data[indexToModify].Address);
                        Console.Write("New Address: ");
                        data[indexToModify].Address = Console.ReadLine();

                        //Extra line for formatting.
                        Console.WriteLine();

                        //Sort the data set after changing the address since address is the
                        //secondary sort criteria
                        data.SortByName();

                        break;

                    case modifyMenu.CITY:
                        //Change the city of the item.
                        Console.WriteLine("Current City: {0}", data[indexToModify].City);
                        Console.Write("New City: ");
                        data[indexToModify].City = Console.ReadLine();

                        //Extra line for formatting.
                        Console.WriteLine();

                        break;

                    case modifyMenu.STATE:
                        //Change the state of the item.
                        Console.WriteLine("Current State: {0}", data[indexToModify].State);
                        Console.Write("New State: ");
                        data[indexToModify].State = Console.ReadLine();

                        //Extra line for formatting.
                        Console.WriteLine();

                        break;

                    case modifyMenu.ZIP:
                        //Change the ZIP code of the item.
                        Console.WriteLine("Current ZIP Code: {0}", data[indexToModify].ZIPCode);
                        Console.Write("New ZIP Code: ");
                        data[indexToModify].ZIPCode = Console.ReadLine();

                        //Extra line for formatting.
                        Console.WriteLine();

                        break;

                    case modifyMenu.PHONE:
                        //Change the phone number of the item.
                        Console.WriteLine("Current Phone Number: {0}", data[indexToModify].PhoneNumber);
                        Console.Write("New Phone Number: ");
                    data[indexToModify].PhoneNumber = Console.ReadLine();

                    //Extra line for formatting.
                    Console.WriteLine();

                    break;

                case modifyMenu.BACK:
                    //Nothing to do; the user wants to go back.
                default:
                    //Catch-all.
                    break;
            }

            //Return choice so the calling method knows what the choice was and can act accordingly.
            return choice;
        }

        /*-------------------------------------------------------------------------------------------
         * Method:  DataSearch
         * Purpose: Interactively searches for objects based upon user input.
         * Input:   Nothing.
         * Output:  Nothing.
        -------------------------------------------------------------------------------------------*/
        private void DataSearch()
        {
            //Display the user's choice.
            Console.WriteLine("----------------");
            Console.WriteLine("| Search Items |");
            Console.WriteLine("----------------");
            Console.Write("Enter your search text: ");

            //Get the user's search text and pipe that directly into the search method.
            CViewDataSet foundData = data.Search(Console.ReadLine());

            //Show the number of items found.
            Console.WriteLine("{0} item{1} found.\n", foundData.Count, foundData.Count == 1 ? "" : "s")↙
    ;

            //If any items found, display them.
            if (foundData.Count != 0)
                Console.WriteLine("{0}\n{1}", foundData.Header, foundData);
        }

        /*-------------------------------------------------------------------------------------------
         * Method:  DataDelete
         * Purpose: Interactively deletes an object based upon user input.
         * Input:   Nothing.
         * Output:  Nothing.
        -------------------------------------------------------------------------------------------*/
        private void DataDelete()
        {
            //Display the user's choice.
            Console.WriteLine("----------------------------------");
            Console.WriteLine("| Delete Item -- Existing Items |");
            Console.WriteLine("----------------------------------");

            //Display a numbered list of all the objects in the data set.
            DataDisplayAllNumbered();

            //Get the user's choice of which object to delete.
            Console.Write("\nSelect item (0 for none): ");
            int indexToDelete = int.Parse(Console.ReadLine()) - 1;

            //Extra line for formatting.
            Console.WriteLine();

            //Validate the user's choice.
            if (indexToDelete == -1)
            {
                //The user changed their mind.
                Console.WriteLine("Cancelled.\n");
                return;
            }
            else if (indexToDelete < 0 || indexToDelete >= data.Count)
            {
                //The user input an invalid object index.
                Console.WriteLine("Invalid item.\n");
                return;
            }

            //Delete the object and display confirmation of its deletion.
            data.Delete(indexToDelete);
            Console.WriteLine("Item {0} has been deleted.\n", indexToDelete + 1);
        }

        /*---------------------------------------------------------------------------------------------
         * Method:  DataDisplayAll
         * Purpose: Displays the header and the serialized dataset object.
         * Input:   Nothing.
         * Output:  Nothing.
        ---------------------------------------------------------------------------------------------*/
        private void DataDisplayAll()
        {
            //Display the user's choice.
            Console.WriteLine("---------------------");
            Console.WriteLine("| Display All Items |");
            Console.WriteLine("---------------------");

            //Display all the objects.
            Console.WriteLine("{0}\n{1}", data.Header, data);
        }

        /*---------------------------------------------------------------------------------------------
         * Method:  DataDisplayAllNumbered
         * Purpose: Display a header and a numbered list of objects.
         * Input:   Nothing.
         * Output:  Nothing.
        ---------------------------------------------------------------------------------------------*/
        private void DataDisplayAllNumbered()
        {
            //Display the header.
            Console.WriteLine("Item  {0}", data.Header);

            //Display the numbered objects, starting at 1.
            for (int objectNum = 0; objectNum < data.Count; objectNum++)
                Console.WriteLine("{0,4}  {1}", objectNum + 1, data[objectNum]);
        }
    }
}

```