```
1 /*-----
2 * Name: Dan Cassidy
3 * Date: 2015-06-02
4 * Assignment: cView-P1
5 * Source File: Program.cs
6 * Class: CSCI-C 490, C# Programming, MoWe 08:00
7 * Purpose: Small wrapper program for demonstrating the CViewDataInteractive class.
8 -----*/
9
10 using System;
11 using System.Collections.Generic;
12 using System.Linq;
13 using System.Text;
14 using System.Threading.Tasks;
15
16 namespace cView_P1_DanCassidy
17 {
     class Program
18
19
     {
        /*-----
20
         * Method: Main
21
         * Purpose: Serves as the entry point to the program.
22
23
         * Input: String array object representing any command line arguments.
24
        * Output: Nothing.
                       */-----*/
25
        static void Main(string[] args)
26
27
           //Declare a new CViewDataInteractive object.
28
29
           var data = new CViewDataInteractive();
30
31
           //Interactively manupulate said object.
32
           data.InteractiveManipulation();
33
        }
34
     }
35 }
36
```

```
C:\Users\Dan\Box Sync\2014-2015 Summer\CSCI-C 490 ...-P1-DanCassidy\cView-P1-DanCassidy\CViewData.cs 1
```

```
1 /*-----
 2 * Name:
               Dan Cassidy
                2015-06-02
 3 * Date:
 4 * Assignment: cView-P1
   * Source File: CViewData.cs
 6 * Class:
                CSCI-C 490, C# Programming, MoWe 08:00
 7 * Purpose:
                Contains the basic class for the cView program, along with some supporting methods.
 9
10 using System;
11 using System.Collections.Generic;
12 using System.Linq;
13 using System.Text;
14 using System.Threading.Tasks;
15
16 namespace cView_P1_DanCassidy
17 {
18
      class CViewData
19
      {
         //Basic properties of the class.
20
         public string Name { get; set; }
21
         public string Address { get; set; }
22
23
          public string City { get; set; }
          public string State { get; set; }
24
25
          public string ZIPCode { get; set; }
         public string PhoneNumber { get; set; }
26
27
         //Easily accessible string showing the data order in the ToString() method.
28
29
         private const string HEADER = "Business Name, Address, City, State, ZIP Code [Phone Number]";
30
31
         //Read-only accessor for the Header property that just uses the HEADER constant.
32
         public static string Header
33
          {
34
             get
35
             {
36
                 return HEADER;
37
             }
         }
39
          /*-----
40
          * Method: Contains
41
42
           * Purpose: Search this object for a string, optionally with case sensitivity.
43
           * Input:
                    string to Search For, representing the string that will be searched for.
                    (Optional) bool caseInsensitive, determines whether the search will be case
44
45
                    sensitive or case insensitive. Default is case insensitive.
           ^{st} Output: bool representing whether the specified string was found in the object.
46
47
          -----*/
48
         public bool Contains(string toSearchFor, bool caseInsensitive = true)
49
50
             //Determine whether to use case sensitive or insensitive searching.
51
             switch (caseInsensitive)
52
             {
                 case false:
53
54
                    //Case sensitive searching.
                    if (Name.Contains(toSearchFor) || Address.Contains(toSearchFor) ||
55
                        City.Contains(toSearchFor) | State.Contains(toSearchFor) |
56
57
                        ZIPCode.Contains(toSearchFor) || PhoneNumber.Contains(toSearchFor))
58
                    {
59
                        //Found it.
60
                        return true;
61
62
                    break;
63
64
                 case true:
65
                 default:
                    //Case insensitive searching. Basic code idea from Stack Overflow.
66
```

```
//http://stackoverflow.com/a/444818
67
                     if (Name.IndexOf(toSearchFor, StringComparison.OrdinalIgnoreCase) >= 0 ||
68
                         Address.IndexOf(toSearchFor, StringComparison.OrdinalIgnoreCase) >= 0 ||
69
70
                         City.IndexOf(toSearchFor, StringComparison.OrdinalIgnoreCase) >= 0 ||
71
                         State.IndexOf(toSearchFor, StringComparison.OrdinalIgnoreCase) >= 0 ||
                         ZIPCode.IndexOf(toSearchFor, StringComparison.OrdinalIgnoreCase) >= 0 ||
72
                         PhoneNumber.IndexOf(toSearchFor, StringComparison.OrdinalIgnoreCase) >= 0)
73
74
75
                         //Found it.
76
                         return true;
77
78
                     break;
79
              }
80
              //If the specified string cannot be found in this object, return false.
81
              return false;
82
83
          }
84
          /*-----
           * Method: ToString
86
           * Purpose: Override of the ToString() method. Formats the return value so it looks pretty.
87
           * Input: Nothing
88
89
           * Output: String object containing serialized object data.
90
                              ._____*/
91
          public override string ToString()
92
              return Name + ", " + Address + ", " + City + ", " + State + ", " + ZIPCode + " [" + PhoneNumber + "]";
93
94
95
          }
96
      }
97 }
98
```

```
1 /*-----
 2 * Name:
                  Dan Cassidv
 3
   * Date:
                  2015-06-02
   * Assignment: cView-P1
   * Source File: CViewDataSet.cs
   * Class:
                  CSCI-C 490, C# Programming, MoWe 08:00
 6
   * Purpose:
 7
                  Builds a List-based class for collections of CViewData objects and contains related
 8 *
                  methods and properties.
 9 ---
10
11 using System;
12 using System.Collections.Generic;
13 using System.Linq;
14 using System.Text;
15 using System.Threading.Tasks;
16
17 namespace cView_P1_DanCassidy
18 {
19
      class CViewDataSet
20
21
           //Basic property of the class.
22
          private List<CViewData> dataSet = new List<CViewData>();
23
24
          //Enable read-only access to the Count property.
25
          public int Count
26
          {
27
              get
28
              {
29
                  return dataSet.Count;
30
              }
          }
31
32
33
          //Enable read-only access to the Header property. Uses the header from the CViewData class
34
          //so if needs to be changed, it only needs to be changed in one place.
35
          public string Header
36
          {
37
              get
38
              {
39
                  return CViewData.Header;
40
              }
41
          }
42
43
44
           * Method: this[]
           ^{st} Purpose: Access the objects in this dataset via index number.
45
           * Input: int objectNum, the index of the object that will be accessed.
46
47
           * Output: CViewData object of the referenced object at the index.
48
49
          public CViewData this[int objectNum]
50
          {
51
              get
52
              {
                  //Try to simply return the object at index objectNum.
53
54
                  try
55
                  {
56
                      return dataSet[objectNum];
57
                  }
                  catch (ArgumentOutOfRangeException)
58
59
                  {
                      //If this exception is caught, let the user know and return a null.
60
                      Console.WriteLine("Index [{0}] is out of range.", objectNum);
61
                      return null;
62
63
                  }
64
              }
65
              set
66
              {
```

```
67
                 //Try to set the object at index objectNum.
68
                 try
 69
                 {
                    dataSet[objectNum] = value;
 70
 71
                 }
                 catch (ArgumentOutOfRangeException)
 72
 73
 74
                    //If this exception is caught, do nothing further and let the user know.
 75
                    Console.WriteLine("Index [{0}] is out of range.", objectNum);
 76
                 }
 77
             }
 78
          }
 79
 80
81
           * Method: Add
           * Purpose: Add a data object to the dataset.
82
83
           * Input: CViewData toAdd, this is the object that will get added to the dataset.
84
           * Output: Nothing.
                           -----*/
85
86
          public void Add(CViewData toAdd)
87
          {
88
             //Add object using List Add method.
 89
             dataSet.Add(toAdd);
90
          }
91
          /*-----
92
           * Method: Delete
93
           * Purpose: Delete an object at the given index from the dataset.
95
           * Input: int indexToRemove, the index of the object to be removed from the dataset.
96
           * Output: Nothing.
97
                                -----*/
98
          public void Delete(int indexToRemove)
99
          {
100
             //Delete object at specified index by using List RemoveAt method.
101
             dataSet.RemoveAt(indexToRemove);
102
          }
103
          /*-----
104
105
           * Method: SortByName
           * Purpose: Sort the dataset by the Name property of the objects, with a secondary sort by
106
107
                    the Address property.
108
           * Input: Nothing.
109
           * Output: Nothing.
          -----*/
110
111
          public void SortByName()
112
             //Idea from Stack Overflow: http://stackoverflow.com/a/3309230
113
114
             //Yay lambda expressions!
115
             dataSet = dataSet.OrderBy(data => data.Name).OrderBy(data => data.Address).ToList();
          }
116
117
118
           * Method: Search
119
120
           * Purpose: Search for a given string in this dataset.
           * Input: string toSearchFor, this is the string that will be searched for.
121
           * Output: CViewDataSet object, containing all (if any) objects found.
122
123
124
          public CViewDataSet Search(string toSearchFor)
125
          {
             //Create a new dataset to hold the found objects.
126
             CViewDataSet foundDataSet = new CViewDataSet();
127
128
129
             //Iterate through the objects and add them to foundDataSet if applicable.
130
             foreach (CViewData data in dataSet)
131
                 if (data.Contains(toSearchFor))
132
                    foundDataSet.Add(data);
```

```
133
134
             //Return the dataset containing the found objects.
135
             return foundDataSet;
136
         }
137
         /*-----
138
          * Method: ToString
139
140
          * Purpose: Override of the ToString() method. Formats the return value so it looks pretty.
          * Input: Nothing.
141
          * Output: String object containing serialized collection data.
142
143
144
         public override string ToString()
145
         {
146
             //Declare the string.
             string toReturn = "";
147
148
             //Build the string.
149
150
             foreach (CViewData item in dataSet)
151
                 toReturn += item.ToString() + "\n";
152
             //Return the string.
153
154
             return toReturn;
155
         }
      }
156
157 }
158
```

```
1 /*-----
2 * Name:
               Dan Cassidy
3 * Date:
                2015-06-02
4 * Assignment: cView-P1
  * Source File: CViewDataInteractive.cs
6 * Class:
                CSCI-C 490, C# Programming, MoWe 08:00
7 * Purpose:
               Provides interactive management of a CViewDataSet object.
10 using System;
11 using System.Collections.Generic;
12 using System.Linq;
13 using System.Text;
14 using System.Threading.Tasks;
15
16 namespace cView_P1_DanCassidy
17 {
18
      class CViewDataInteractive
19
      {
20
         private CViewDataSet data = new CViewDataSet();
21
22
         //Helper constants for menu validation.
23
         private const mainMenu MAINMENU MIN = mainMenu.ADD;
24
         private const mainMenu MAINMENU_MAX = mainMenu.EXIT;
25
         private const modifyMenu MODIFYMENU_MIN = modifyMenu.NAME;
         private const modifyMenu MODIFYMENU_MAX = modifyMenu.BACK;
26
27
         //Enum for the main menu. Basic code idea from Stack Overflow.
28
29
         //http://stackoverflow.com/a/15752719
30
         private enum mainMenu
31
32
             ADD = 1,
33
             MODIFY,
34
             SEARCH,
35
             DELETE,
36
             DISPLAY_ALL,
37
             EXIT
38
         }
39
40
         //Enum for the modify menu. Basic code idea from Stack Overflow.
41
         //http://stackoverflow.com/a/15752719
42
         private enum modifyMenu
43
         {
44
             NAME = 1,
45
             ADDRESS,
46
             CITY,
47
             STATE,
48
             ZIP,
49
             PHONE,
50
             BACK
51
         }
52
53
          /*-----
54
          * Method: InteractiveManipulation
          * Purpose: Entry point for interactive manipulation of CViewDataSet object.
55
          * Input: Nothing.
56
57
          * Output: Nothing.
                              ----*/
58
59
         public void InteractiveManipulation()
60
             //Loop the main menu until the user decides to exit.
61
62
             while (MainMenuAction(MainMenuDisplay()) != mainMenu.EXIT) ;
63
         }
64
65
          * Method: MainMenuDisplay
66
```

```
67
            * Purpose: Display the main menu and get a choice. Must have valid input to return.
68
            * Input: Nothing.
 69
            * Output: mainMenu, representing the choice that was made.
 70
 71
           private mainMenu MainMenuDisplay()
 72
 73
               mainMenu menuChoice;
 74
               bool invalid;
 75
 76
               do
 77
               {
 78
                   //Display the menu.
                   Console.WriteLine("-----");
 79
                  Console.WriteLine("| Main Interactive Menu |");
Console.WriteLine("-----");
 80
81
                   Console.WriteLine("Please select an option:");
82
                   Console.WriteLine(" 1) Add New Item");
83
                   Console.WriteLine(" 2) Modify Item");
84
                   Console.WriteLine(" 3) Search Items");
85
                   Console.WriteLine(" 4) Delete Item");
86
                   Console.WriteLine(" 5) Display All Items");
87
                   Console.WriteLine(" 6) Exit");
88
89
                  Console.Write("Choice: ");
90
91
                   //Get the user's choice.
92
                   string input = Console.ReadLine();
93
                   //Extra line for formatting.
95
                   Console.WriteLine();
96
97
                   //Validate the user input.
98
                   invalid = !mainMenu.TryParse(input, out menuChoice) ||
99
                            !MainMenuValidate(menuChoice);
100
               } while (invalid);
101
102
               //Return the user's choice.
103
               return menuChoice;
104
           }
105
           /*-----
106
            * Method: MainMenuValidate
107
108
            * Purpose: Validates that the choice by the user is within the limits and is logically
                      possible.
109
110
                      mainMenu value, contains the user's choice.
            \ensuremath{^{*}} Output: bool, representing whether the user's choice was valid or not.
111
           */----*/
112
           private bool MainMenuValidate(mainMenu value)
113
114
           {
115
               //Check to make sure that the user input is within valid limits.
116
               if (value < MAINMENU_MIN || value > MAINMENU_MAX)
117
                   return false;
118
               //If the data set is empty, limit user to adding an entry or exiting.
119
120
               if (data.Count == 0 && (value != mainMenu.ADD && value != mainMenu.EXIT))
121
               {
122
                   Console.WriteLine("No data is present. Please choose a different option.\n");
123
                   return false;
124
125
               //Otherwise, input is good.
126
127
               return true;
128
           }
129
130
            * Method: MainMenuAction
131
            * Purpose: Acts on the user's choice made at the Main Menu.
132
```

```
133
            * Input:
                     mainMenu choice, represents the action specified.
            * Output: mainMenu, represents the action specified.
134
135
                                                              ----*/
136
          private mainMenu MainMenuAction(mainMenu choice)
137
              //Decide what to do based on the user's choice.
138
139
              switch (choice)
140
                  case mainMenu.ADD:
141
142
                     //Add a new item.
143
                     DataAdd();
144
                     break;
145
146
                  case mainMenu.MODIFY:
147
                      //Modify an existing item.
148
                     DataModify();
149
                     break;
150
                  case mainMenu.SEARCH:
151
152
                     //Search items.
153
                     DataSearch();
154
                     break;
155
156
                  case mainMenu.DELETE:
157
                      //Delete an item.
158
                     DataDelete();
159
                     break;
160
                  case mainMenu.DISPLAY_ALL:
161
162
                      //Display all the items.
163
                     DataDisplayAll();
164
                     break;
165
166
                  case mainMenu.EXIT:
167
                      //Do nothing, exiting the method.
168
                  default:
169
                     //Catch-all.
170
                     break;
171
              }
172
              //Return choice so the calling method knows what the choice was and can act accordingly.
173
174
              return choice;
175
          }
176
           /*-----
177
           * Method: DataAdd
178
179
           * Purpose: Interactively add an item based on the user's input.
180
           * Input: Nothing.
           * Output: Nothing.
181
                               -----*/
182
183
           private void DataAdd()
184
185
              CViewData tempData = new CViewData();
186
187
              //Prompt the user to input information about the new item.
              Console.WriteLine("----");
188
              Console.WriteLine(" | Add New Item |");
189
              Console.WriteLine("----");
190
191
              Console.Write("Business Name: ");
192
              tempData.Name = Console.ReadLine();
              Console.Write("Address: ");
193
194
              tempData.Address = Console.ReadLine();
195
              Console.Write("City: ");
196
              tempData.City = Console.ReadLine();
              Console.Write("State: ");
197
198
              tempData.State = Console.ReadLine();
```

```
Console.Write("ZIP Code: ");
199
              tempData.ZIPCode = Console.ReadLine();
200
              Console.Write("Phone Number: ");
201
202
              tempData.PhoneNumber = Console.ReadLine();
203
204
205
              Console.WriteLine();
206
207
              //Add the new item to the main data set.
208
              data.Add(tempData);
209
210
              //Sort the data set.
211
              data.SortByName();
212
          }
213
214
           * Method: DataModify
215
216
           * Purpose: Interactively modifies an object based on the user's input.
           * Input: Nothing.
           * Output: Nothing.
218
                              -----*/
219
          private void DataModify()
220
221
222
              //Display the user's choice.
              Console.WriteLine("----");
223
              Console.WriteLine(" | Modify Item -- Existing Items | ");
224
              Console.WriteLine("-----");
225
226
              //Display a numbered list of all the objects in the data set.
227
228
              DataDisplayAllNumbered();
229
              //Get the user's choice of which object to delete.
230
              Console.Write("\nSelect item (0 for none): ");
231
              int indexToModify = int.Parse(Console.ReadLine()) - 1;
232
233
234
              //Extra line for formatting.
235
              Console.WriteLine();
236
237
              //Validate the user's choice.
238
              if (indexToModify == -1)
239
240
                  //The user changed their mind.
                  Console.WriteLine("Cancelled.\n");
241
242
                  return;
243
              else if (indexToModify < 0 || indexToModify >= data.Count)
244
245
246
                  //The user input an invalid object index.
247
                  Console.WriteLine("Invalid item.\n");
248
                  return;
249
              }
250
              do
251
252
              {
253
                  //Display the chosen object.
                  Console.WriteLine("-----");
254
                  Console.WriteLine("| Modify Item -- Chosen Item |");
255
                  Console.WriteLine("-----");
256
                  Console.WriteLine("{0}\n{1}\n", data.Header, data[indexToModify]);
257
258
                  //Loop while the use has not chosen to go back.
259
260
              } while (ModifyMenuAction(ModifyMenuDisplay(), indexToModify) != modifyMenu.BACK);
261
          }
262
          /*-----
263
           * Method: ModifyMenuDisplay
264
```

```
265
            * Purpose: Display the modify menu and get a choice. Must have valid input to return.
            * Input: Nothing.
266
267
            * Output: modifyMenu, representing the choice that was made.
268
          private modifyMenu ModifyMenuDisplay()
269
270
              modifyMenu menuChoice;
271
272
              bool invalid;
273
274
              do
275
              {
276
                  //Display the menu.
277
                  Console.WriteLine("Please select the field you would like to modify:");
                  Console.WriteLine(" 1) Business Name");
Console.WriteLine(" 2) Street Address");
278
279
                  Console.WriteLine(" 3) City");
280
                  Console.WriteLine(" 4) State");
281
                  Console.WriteLine(" 5) ZIP Code");
282
                  Console.WriteLine(" 6) Phone Number");
283
                  Console.WriteLine(" 7) Back");
284
285
                  Console.Write("Choice: ");
286
287
                  //Get the user's choice.
288
                  string input = Console.ReadLine();
289
                  //Extra line for formatting.
290
291
                  Console.WriteLine();
292
293
                  //Validate the user input.
294
                  invalid = !modifyMenu.TryParse(input, out menuChoice) ||
                           !ModifyMenuValidate(menuChoice);
295
296
              } while (invalid);
297
298
              //Return the user's choice.
299
              return menuChoice;
300
          }
301
           /*-----
302
303
           * Method: ModifyMenuValidate
           ^{st} Purpose: Validates that the choice by the user is within the limits and is logically
304
305
                     possible.
306
           * Input:
                     mmodifyMenu value, contains the user's choice.
307
           * Output: bool, representing whether the user's choice was valid or not.
308
           -----*/
309
          private bool ModifyMenuValidate(modifyMenu value)
310
              //Check to make sure that the user input is within valid limits.
311
312
              if (value < MODIFYMENU_MIN || value > MODIFYMENU_MAX)
313
                  return false;
314
315
              //Otherwise, input is good.
316
              return true;
317
          }
318
           /*-----
319
            * Method: ModifyMenuAction
320
321
           * 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.
323
324
          private modifyMenu ModifyMenuAction(modifyMenu choice, int indexToModify)
325
326
327
              //Decide what to do based on the user's choice.
328
              switch (choice)
329
                  case modifyMenu.NAME:
330
```

```
331
                        //Change the name of the item.
                        Console.WriteLine("Current Business Name: {0}", data[indexToModify].Name);
332
333
                        Console.Write("New Business Name: ");
334
                        data[indexToModify].Name = Console.ReadLine();
335
                        //Extra line for formatting.
336
337
                        Console.WriteLine();
338
339
                        //Sort the data set after changing the name since name is the primary
340
                        //sort criteria.
341
                        data.SortByName();
342
343
                        break;
344
345
                    case modifyMenu.ADDRESS:
                        //Change the address of the item.
346
347
                        Console.WriteLine("Current Address: {0}", data[indexToModify].Address);
348
                        Console.Write("New Address: ");
349
                        data[indexToModify].Address = Console.ReadLine();
350
351
                        //Extra line for formatting.
352
                        Console.WriteLine();
353
354
                        //Sort the data set after changing the address since address is the
355
                        //secondary sort criteria
356
                        data.SortByName();
357
358
                        break;
359
360
                    case modifyMenu.CITY:
361
                        //Change the city of the item.
362
                        Console.WriteLine("Current City: {0}", data[indexToModify].City);
                        Console.Write("New City: ");
363
364
                        data[indexToModify].City = Console.ReadLine();
365
366
                        //Extra line for formatting.
367
                        Console.WriteLine();
368
369
                        break;
370
                    case modifyMenu.STATE:
371
                        //Change the state of the item.
372
373
                        Console.WriteLine("Current State: {0}", data[indexToModify].State);
                        Console.Write("New State: ");
374
375
                        data[indexToModify].State = Console.ReadLine();
376
377
                        //Extra line for formatting.
378
                        Console.WriteLine();
379
                        break;
380
381
382
                    case modifyMenu.ZIP:
383
                        //Change the ZIP code of the item.
384
                        Console.WriteLine("Current ZIP Code: {0}", data[indexToModify].ZIPCode);
                        Console.Write("New ZIP Code: ");
385
                        data[indexToModify].ZIPCode = Console.ReadLine();
386
387
388
                        //Extra line for formatting.
                        Console.WriteLine();
389
390
391
                        break;
392
393
                    case modifyMenu.PHONE:
394
                        //Change the phone number of the item.
                        Console.WriteLine("Current Phone Number: {0}", data[indexToModify].PhoneNumber);
395
                        Console.Write("New Phone Number: ");
396
```

```
C:\Users\Dan\Box Sync\2014-2015 Summer\CSCI-C 490...\cView-P1-DanCassidy\CViewDataInteractive.cs
397
                     data[indexToModify].PhoneNumber = Console.ReadLine();
398
399
                     //Extra line for formatting.
400
                     Console.WriteLine();
401
402
                     break;
403
404
                 case modifyMenu.BACK:
405
                     //Nothing to do; the user wants to go back.
406
                 default:
407
                     //Catch-all.
408
                     break;
409
              }
410
411
              //Return choice so the calling method knows what the choice was and can act accordingly.
412
              return choice;
          }
413
414
          /*-----
415
416
           * Method: DataSearch
           * Purpose: Interactively searches for objects based upon user input.
417
           * Input: Nothing.
418
419
           * Output: Nothing.
420
                           -----*/
421
          private void DataSearch()
422
423
              //Display the user's choice.
              Console.WriteLine("----");
424
              Console.WriteLine("| Search Items |");
425
426
              Console.WriteLine("----");
              Console.Write("Enter your search text: ");
427
428
429
              //Get the user's search text and pipe that directly into the search method.
430
              CViewDataSet foundData = data.Search(Console.ReadLine());
431
432
              //Show the number of items found.
              Console.WriteLine("{0} item{1} found.\n", foundData.Count, foundData.Count == 1 ? "" : "s") ✔
433
       ;
434
435
              //If any items found, display them.
              if (foundData.Count != 0)
436
437
                 Console.WriteLine("{0}\n{1}", foundData.Header, foundData);
438
          }
439
440
           * Method: DataDelete
441
           * Purpose: Interactively deletes an object based upon user input.
442
443
           * Input: Nothing.
444
           * Output: Nothing.
                                  -----*/
445
446
          private void DataDelete()
447
              //Display the user's choice.
448
449
              Console.WriteLine("-----");
              Console.WriteLine("| Delete Item -- Existing Items |"
450
              Console.WriteLine("-----");
451
452
              //Display a numbered list of all the objects in the data set.
453
454
              DataDisplayAllNumbered();
455
              //Get the user's choice of which object to delete.
456
457
              Console.Write("\nSelect item (0 for none): ");
```

int indexToDelete = int.Parse(Console.ReadLine()) - 1;

//Extra line for formatting.

Console.WriteLine();

458

459

460

461

```
462
             //Validate the user's choice.
463
             if (indexToDelete == -1)
464
465
             {
466
                 //The user changed their mind.
                 Console.WriteLine("Cancelled.\n");
467
                 return;
468
469
             else if (indexToDelete < 0 || indexToDelete >= data.Count)
470
471
                 //The user input an invalid object index.
472
473
                 Console.WriteLine("Invalid item.\n");
474
                 return;
475
476
477
             //Delete the object and display confirmation of its deletion.
478
             data.Delete(indexToDelete);
479
             Console.WriteLine("Item {0} has been deleted.\n", indexToDelete + 1);
480
          }
481
          /*-----
482
           * Method: DataDisplayAll
483
484
           * Purpose: Displays the header and the serialized dataset object.
           * Input: Nothing.
485
           * Output: Nothing.
486
          */----*/
487
488
          private void DataDisplayAll()
489
490
             //Display the user's choice.
             Console.WriteLine("-----");
491
             Console.WriteLine("| Display All Items |");
492
493
             Console.WriteLine("-----");
494
495
             //Display all the objects.
             Console.WriteLine("{0}\n{1}", data.Header, data);
496
497
          }
498
          /*-----
499
500
           * Method: DataDisplayAllNumbered
           * Purpose: Display a header and a numbered list of objects.
501
502
           * Input: Nothing.
503
           * Output: Nothing.
504
505
          private void DataDisplayAllNumbered()
506
             //Display the header.
507
508
             Console.WriteLine("Item {0}", data.Header);
509
510
             //Display the numbered objects, starting at 1.
             for (int objectNum = 0; objectNum < data.Count; objectNum++)</pre>
511
512
                 Console.WriteLine("{0,4} {1}", objectNum + 1, data[objectNum]);
513
          }
514
      }
515 }
516
```

Main Interactive Menu and Add New Item

```
| Main Interactive Menu !
| Please select an option:
| 1) Add New Item
| 3) Search Items
| 4) Delete Item
| 5) Display All Items
| 6) Exit |
| Rudd New Item !
| Business Name: Econo Lodge |
| Add New Item |
| Business Name: Econo Lodge |
| Address: 3233 Lincoln Way W |
| City: South Bend |
| State: IN |
| ZIP Code: 46628 |
| Phone Number: 574-232-9019 |
| Main Interactive Menu !
```

Cancelling and then attempting invalid input in Modify Item

Modify Item and Modify Item Menu

```
🔳 file:///C:/Users/Dan/Box Sync/2014-2015 Summer/CSCI-C 490 (C# Programming)/Project/Phase 1/cView-P1-DanCassidy/cView-...
   Main Interactive Menu !
Please select an option:

1) Add New Item
2) Modify Item
3) Search Items
4) Delete Item
5) Display All Items
6) Exit
Choice: 2
| Modify Item -- Existing Items |
Item Business Name, Address, City, State, ZIP Code [Phone Number]
1 Econo Lodge, 3233 Lincoln Way W, South Bend, IN, 46628 [574-232-9019]
Select item (0 for none): 1
| Modify Item -- Chosen Item |
Business Name, Address, City, State, ZIP Code [Phone Number]
Econo Lodge, 3233 Lincoln Way W, South Bend, IN, 46628 [574–232–9019]
Please select the field you would like to modify:

1) Business Name
2) Street Address
3) City
4) State
5) ZIP Code
6) Phone Number
7) Back
Choice: 4
Current State: IN
New State: NI
| Modify Item -- Chosen Item |
Business Name, Address, City, State, ZIP Code [Phone Number]
Econo Lodge, 3233 Lincoln Way W, South Bend, NI, 46628 [574–232–9019]
Please select the field you would like to modify:
1) Business Name
2) Street Address
3) City
4) State
5) ZIP Code
6) Phone Number
7) Back
Choice: 7
| Main Interactive Menu |
```

Searching and items found

Searching and no items found

Display All Items (Pre-Delete)

Cancelling and then attempting invalid input in Delete Item

Delete Item

```
| Main Interactive Menu |
| Please select an option:
| 1) Add New Item
| 2) Modify Item
| 3) Search Items
| 4) Delete Item -- Existing Items |
| Item Business Name, Address, City, State, ZIP Code [Phone Number]
| 1 Ashton Mechanical, Inc, 1 Out of Area Ave, South Bend, IN, 46601 [574-291-7732]
| 2 Econo Lodge, 3233 Lincoln Way W, South Bend, IN, 46628 [574-232-9019]
| Select item (0 for none): 1
| Item 1 has been deleted.
| Main Interactive Menu |
```

Display All Items (Post-Delete)

Attempting to modify an item when empty.