

```
1 /*-----
2  * Name:      Dan Cassidy
3  * Date:      2015-06-02
4  * Assignment: cView-P1
5  * Source File: CViewDataInteractive.cs
6  * Class:     CSCI-C 490, C# Programming, MoWe 08:00
7  * Purpose:   Provides interactive management of a CViewDataSet object.
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 {
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;
26         private const modifyMenu MODIFYMENU_MAX = modifyMenu.BACK;
27
28         //Enum for the main menu. Basic code idea from Stack Overflow.
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
55         * Purpose: Entry point for interactive manipulation of CViewDataSet object.
56         * Input:   Nothing.
57         * Output:  Nothing.
58         *-----*/
59         public void InteractiveManipulation()
60         {
61             //Loop the main menu until the user decides to exit.
62             while (MainMenuAction(MainMenuDisplay()) != mainMenu.EXIT) ;
63         }
64
65         /*-----
66         * Method: MainMenuDisplay
```

```
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.
79          Console.WriteLine("-----");
80          Console.WriteLine("| Main Interactive Menu |");
81          Console.WriteLine("-----");
82          Console.WriteLine("Please select an option:");
83          Console.WriteLine(" 1) Add New Item");
84          Console.WriteLine(" 2) Modify Item");
85          Console.WriteLine(" 3) Search Items");
86          Console.WriteLine(" 4) Delete Item");
87          Console.WriteLine(" 5) Display All Items");
88          Console.WriteLine(" 6) Exit");
89          Console.Write("Choice: ");
90
91          //Get the user's choice.
92          string input = Console.ReadLine();
93
94          //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  /*-----
107  * Method:  MainMenuValidate
108  * Purpose: Validates that the choice by the user is within the limits and is logically
109  *           possible.
110  * Input:   mainMenu value, contains the user's choice.
111  * Output:  bool, representing whether the user's choice was valid or not.
112  -----*/
113  private bool MainMenuValidate(mainMenu value)
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
119      //If the data set is empty, limit user to adding an entry or exiting.
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
126      //Otherwise, input is good.
127      return true;
128  }
129
130  /*-----
131  * Method:  MainMenuAction
132  * Purpose: Acts on the user's choice made at the Main Menu.
```

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

```

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

```

```
265     * Purpose: Display the modify menu and get a choice. Must have valid input to return.
266     * Input:   Nothing.
267     * Output:  modifyMenu, representing the choice that was made.
268     -----*/
269 private modifyMenu ModifyMenuDisplay()
270 {
271     modifyMenu menuChoice;
272     bool invalid;
273
274     do
275     {
276         //Display the menu.
277         Console.WriteLine("Please select the field you would like to modify:");
278         Console.WriteLine(" 1) Business Name");
279         Console.WriteLine(" 2) Street Address");
280         Console.WriteLine(" 3) City");
281         Console.WriteLine(" 4) State");
282         Console.WriteLine(" 5) ZIP Code");
283         Console.WriteLine(" 6) Phone Number");
284         Console.WriteLine(" 7) Back");
285         Console.Write("Choice: ");
286
287         //Get the user's choice.
288         string input = Console.ReadLine();
289
290         //Extra line for formatting.
291         Console.WriteLine();
292
293         //Validate the user input.
294         invalid = !modifyMenu.TryParse(input, out menuChoice) ||
295                 !ModifyMenuValidate(menuChoice);
296     } while (invalid);
297
298     //Return the user's choice.
299     return menuChoice;
300 }
301
302 /*-----
303  * Method:  ModifyMenuValidate
304  * Purpose: Validates that the choice by the user is within the limits and is logically
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 {
311     //Check to make sure that the user input is within valid limits.
312     if (value < MODIFYMENU_MIN || value > MODIFYMENU_MAX)
313         return false;
314
315     //Otherwise, input is good.
316     return true;
317 }
318
319 /*-----
320  * Method:  ModifyMenuAction
321  * Purpose: Acts on the user's choice made at the Modify Menu.
322  * Input:   modifyMenu choice, represents the action specified.
323  * Output:  modifyMenu, represents the action specified.
324  -----*/
325 private modifyMenu ModifyMenuAction(modifyMenu choice, int indexToModify)
326 {
327     //Decide what to do based on the user's choice.
328     switch (choice)
329     {
330         case modifyMenu.NAME:
```

```
331         //Change the name of the item.
332         Console.WriteLine("Current Business Name: {0}", data[indexToModify].Name);
333         Console.Write("New Business Name: ");
334         data[indexToModify].Name = Console.ReadLine();
335
336         //Extra line for formatting.
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:
346         //Change the address of the item.
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);
363         Console.Write("New City: ");
364         data[indexToModify].City = Console.ReadLine();
365
366         //Extra line for formatting.
367         Console.WriteLine();
368
369         break;
370
371     case modifyMenu.STATE:
372         //Change the state of the item.
373         Console.WriteLine("Current State: {0}", data[indexToModify].State);
374         Console.Write("New State: ");
375         data[indexToModify].State = Console.ReadLine();
376
377         //Extra line for formatting.
378         Console.WriteLine();
379
380         break;
381
382     case modifyMenu.ZIP:
383         //Change the ZIP code of the item.
384         Console.WriteLine("Current ZIP Code: {0}", data[indexToModify].ZIPCode);
385         Console.Write("New ZIP Code: ");
386         data[indexToModify].ZIPCode = Console.ReadLine();
387
388         //Extra line for formatting.
389         Console.WriteLine();
390
391         break;
392
393     case modifyMenu.PHONE:
394         //Change the phone number of the item.
395         Console.WriteLine("Current Phone Number: {0}", data[indexToModify].PhoneNumber);
396         Console.Write("New Phone Number: ");
```

```
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
417  * Purpose: Interactively searches for objects based upon user input.
418  * Input:  Nothing.
419  * Output: Nothing.
420 -----*/
421 private void DataSearch()
422 {
423     //Display the user's choice.
424     Console.WriteLine("-----");
425     Console.WriteLine("| Search Items |");
426     Console.WriteLine("-----");
427     Console.Write("Enter your search text: ");
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.
433     Console.WriteLine("{0} item{1} found.\n", foundData.Count, foundData.Count == 1 ? "" : "s") ↵
434 ;
435
436     //If any items found, display them.
437     if (foundData.Count != 0)
438         Console.WriteLine("{0}\n{1}", foundData.Header, foundData);
439 }
440
441 /*-----
442  * Method: DataDelete
443  * Purpose: Interactively deletes an object based upon user input.
444  * Input:  Nothing.
445  * Output: Nothing.
446 -----*/
447 private void DataDelete()
448 {
449     //Display the user's choice.
450     Console.WriteLine("-----");
451     Console.WriteLine("| Delete Item -- Existing Items |");
452     Console.WriteLine("-----");
453
454     //Display a numbered list of all the objects in the data set.
455     DataDisplayAllNumbered();
456
457     //Get the user's choice of which object to delete.
458     Console.Write("\nSelect item (0 for none): ");
459     int indexToDelete = int.Parse(Console.ReadLine()) - 1;
460
461     //Extra line for formatting.
462     Console.WriteLine();
```

```

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

```