

```

1  /*-----
2  * Name:      Dan Cassidy
3  * Date:      2015-06-09
4  * Assignment: cView-P2
5  * Source File: CViewDataInteractive.cs
6  * Course:    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
17 {
18     class CViewDataInteractive
19     {
20         //Helper constants for menu validation.
21         private const mainMenu MAINMENU_MIN = mainMenu.ADD;
22         private const mainMenu MAINMENU_MAX = mainMenu.EXIT;
23         private const fieldMenu FIELDMENU_MIN = fieldMenu.NAME;
24         private const fieldMenu FIELDMENU_MAX = fieldMenu.BACK;
25
26         //Primary class field/instance variable.
27         private CViewDataSet data = new CViewDataSet();
28
29         //Enum for the main menu. Basic code idea from Stack Overflow.
30         //http://stackoverflow.com/a/15752719
31         private enum mainMenu
32         {
33             ADD = 1,
34             MODIFY,
35             SEARCH,
36             DELETE,
37             DISPLAY_ALL,
38             EXIT
39         }
40
41         //Enum for the modify menu. Basic code idea from Stack Overflow.
42         //http://stackoverflow.com/a/15752719
43         private enum fieldMenu
44         {
45             NAME = CViewData.FIELDS_MIN,
46             FACILITY,
47             ADDRESS,
48             CITY,
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:   DataAdd
67     * Purpose: Interactively add an item based on the user's input.
68     * Input:    Nothing.
69     * Output:   Nothing.
70     -----*/
71     private void DataAdd()
72     {
73         //New CViewData object that will be added to the dataset.
74         CViewData dataToAdd = new CViewData();
75
76         //Prompt the user to input information about the new item.
77         Console.WriteLine("-----");
78         Console.WriteLine("| Add New Item |");
79         Console.WriteLine("-----");
80         Console.Write("Facility Name: ");
81         dataToAdd.Name = Console.ReadLine();
82         Console.Write("Facility Type: ");
83         dataToAdd.FacilityType = Console.ReadLine();
84         Console.Write("Address: ");
85         dataToAdd.Address = Console.ReadLine();
86         Console.Write("City: ");
87         dataToAdd.City = Console.ReadLine();
88         Console.Write("Phone Number: ");
89         dataToAdd.PhoneNumber = Console.ReadLine();
90
91         //Extra line for formatting.
92         Console.WriteLine();
93
94         //Add the new item to the main data set.
95         data.Add(dataToAdd);
96
97         //Sort the data set.
98         data.SortByName();
99     }
100
101     /*-----
102     * Method:   DataDelete
103     * Purpose: Interactively deletes an object based upon user input.
104     * Input:    Nothing.
105     * Output:   Nothing.
106     -----*/
107     private void DataDelete()
108     {
109         //Default value of 0 in case the user doesn't enter a choice and just hits 'enter'.
110         int indexToDelete = 0;
111
112         //Display the user's choice.
113         Console.WriteLine("-----");
114         Console.WriteLine("| Delete Item -- Existing Items |");
115         Console.WriteLine("-----");
116
117         //Display a numbered list of all the objects in the data set.
118         DataDisplayAllNumbered();
119
120         //Get the user's choice of which object to delete.
121         Console.Write("\nSelect item (0 to cancel): ");
122         int.TryParse(Console.ReadLine(), out indexToDelete);
123         indexToDelete--;
124
125         //Extra line for formatting.
126         Console.WriteLine();
127
128         //Display the results.

```

```

129         Console.WriteLine("-----");
130         Console.WriteLine(" | Delete Item -- Results |");
131         Console.WriteLine("-----");
132
133         //Validate the user's choice.
134         if (indexToDelete == -1)
135         {
136             //The user changed their mind.
137             Console.WriteLine("Cancelled.\n");
138             return;
139         }
140         else if (indexToDelete < 0 || indexToDelete >= data.Count)
141         {
142             //The user input an invalid object index.
143             Console.WriteLine("Invalid item.\n");
144             return;
145         }
146
147         //Delete the object and display confirmation of its deletion.
148         data.Delete(indexToDelete);
149         Console.WriteLine("Item {0} has been deleted.\n", indexToDelete + 1);
150
151         //Display the still existing items.
152         DataDisplayAll(false);
153     }
154
155     /*-----
156     * Method:   DataDisplayAll
157     * Purpose:  Displays the header and the serialized dataset object.
158     * Input:    bool displayTitle, determines whether the method should print a title showing
159     *           that this method was the one that was called.
160     * Output:   Nothing.
161     -----*/
162     private void DataDisplayAll(bool displayTitle = true)
163     {
164         //Choose whether to display the title.
165         if (displayTitle)
166         {
167             //Display the user's choice.
168             Console.WriteLine("-----");
169             Console.WriteLine(" | Display All Items |");
170             Console.WriteLine("-----");
171         }
172
173         //Display all the objects.
174         Console.WriteLine("{0}\n{1}", data.Header, data.Count != 0 ?
175             data.ToString() : "No items currently stored.\n");
176     }
177
178     /*-----
179     * Method:   DataDisplayAllNumbered
180     * Purpose:  Display a header and a numbered list of objects.
181     * Input:    Nothing.
182     * Output:   Nothing.
183     -----*/
184     private void DataDisplayAllNumbered()
185     {
186         //Display the header.
187         Console.WriteLine("Item {0}", data.Header);
188
189         //If the dataset is not empty.
190         if (data.Count != 0)
191             //Display the numbered objects, starting at 1.
192             for (int objectNum = 0; objectNum < data.Count; objectNum++)

```

```

193         Console.WriteLine("{0,4} {1}", objectNum + 1, data[objectNum]);
194     else
195         //Display a message saying that dataset is empty.
196         Console.WriteLine("No items currently stored.");
197 }
198
199 /*-----
200  * Method:   DataModify
201  * Purpose:  Interactively modifies an object based on the user's input.
202  * Input:    Nothing.
203  * Output:   Nothing.
204  -----*/
205 private void DataModify()
206 {
207     //Default value of 0 in case the user doesn't enter a choice and just hits 'enter'.
208     int indexToModify = 0;
209
210     //Display the user's choice.
211     Console.WriteLine("-----");
212     Console.WriteLine("| Modify Item -- Existing Items |");
213     Console.WriteLine("-----");
214
215     //Display a numbered list of all the objects in the data set.
216     DataDisplayAllNumbered();
217
218     //Get the user's choice of which object to delete.
219     Console.Write("\nSelect item (0 to cancel): ");
220     int.TryParse(Console.ReadLine(), out indexToModify);
221     indexToModify--;
222
223     //Extra line for formatting.
224     Console.WriteLine();
225
226     //Validate the user's choice.
227     if (indexToModify == -1)
228     {
229         //The user changed their mind.
230         Console.WriteLine("Cancelled.\n");
231         return;
232     }
233     else if (indexToModify < 0 || indexToModify >= data.Count)
234     {
235         //The user input an invalid object index.
236         Console.WriteLine("Invalid item.\n");
237         return;
238     }
239
240     do
241     {
242         //Display the chosen object.
243         Console.WriteLine("-----");
244         Console.WriteLine("| Modify Item -- Chosen Item |");
245         Console.WriteLine("-----");
246         Console.WriteLine("{0}\n{1}\n", data.Header, data[indexToModify]);
247
248         //Loop while the use has not chosen to go back.
249     } while (DataModifyMenuAction(FieldMenuDisplay(), indexToModify) != fieldMenu.BACK);
250 }
251
252 /*-----
253  * Method:   DataModifyMenuAction
254  * Purpose:  Acts on the user's choice made at the Modify Menu.
255  * Input:    fieldMenu choice, represents the action specified.
256  * Output:   fieldMenu, represents the action specified.

```

```
257 -----*/
258 private fieldMenu DataModifyMenuAction(fieldMenu choice, int indexToModify)
259 {
260     //Decide what to do based on the user's choice.
261     switch (choice)
262     {
263         case fieldMenu.NAME:
264             //Change the name of the item.
265             Console.WriteLine("Current Facility Name: {0}", data[indexToModify].Name);
266             Console.Write("New Facility Name: ");
267             data[indexToModify].Name = Console.ReadLine();
268
269             //Sort the data set after changing the name since name is the sort criteria.
270             data.SortByName();
271
272             break;
273
274         case fieldMenu.FACILITY:
275             //Change the facility type of the item.
276             Console.WriteLine("Current Facility Type: {0}",
277                             data[indexToModify].FacilityType);
278             Console.Write("New Facility Type: ");
279             data[indexToModify].FacilityType = Console.ReadLine();
280
281             break;
282
283         case fieldMenu.ADDRESS:
284             //Change the address of the item.
285             Console.WriteLine("Current Address: {0}", data[indexToModify].Address);
286             Console.Write("New Address: ");
287             data[indexToModify].Address = Console.ReadLine();
288
289             break;
290
291         case fieldMenu.CITY:
292             //Change the city of the item.
293             Console.WriteLine("Current City: {0}", data[indexToModify].City);
294             Console.Write("New City: ");
295             data[indexToModify].City = Console.ReadLine();
296
297             break;
298
299         case fieldMenu.PHONE:
300             //Change the phone number of the item.
301             Console.WriteLine("Current Phone Number: {0}", data[indexToModify].PhoneNumber);
302             Console.Write("New Phone Number: ");
303             data[indexToModify].PhoneNumber = Console.ReadLine();
304
305             break;
306
307         case fieldMenu.BACK:
308             //Nothing to do; the user wants to go back.
309         default:
310             //Catch-all.
311             return choice;
312     }
313
314     //Extra line for formatting.
315     Console.WriteLine();
316
317     //Return choice so the calling method knows what the choice was and can act accordingly.
318     return choice;
319 }
320
```

```

321     /*-----
322     * Method:  DataSearch
323     * Purpose: Interactively searches for objects based upon user input.
324     * Input:   Nothing.
325     * Output:  Nothing.
326     -----*/
327     private void DataSearch()
328     {
329         do
330         {
331             //Display the user's choice.
332             Console.WriteLine("-----");
333             Console.WriteLine("| Search Items |");
334             Console.WriteLine("-----");
335
336             //Loop while the use has not chosen to go back.
337         } while (DataSearchMenuAction(FieldMenuDisplay()) != fieldMenu.BACK);
338     }
339
340     /*-----
341     * Method:  DataSearchMenuAction
342     * Purpose: Acts on the user's choice made at the Search Menu.
343     * Input:   fieldMenu choice, represents the action specified.
344     * Output:  fieldMenu, represents the action specified.
345     -----*/
346     private fieldMenu DataSearchMenuAction(fieldMenu choice)
347     {
348         //Decide what to display based on the user's choice.
349         switch (choice)
350         {
351             case fieldMenu.NAME:
352                 //Search the name field.
353                 Console.WriteLine("-----");
354                 Console.WriteLine("| Search Items -- Facility Name |");
355                 Console.WriteLine("-----");
356                 break;
357
358             case fieldMenu.FACILITY:
359                 //Search the facility type field.
360                 Console.WriteLine("-----");
361                 Console.WriteLine("| Search Items -- Facility Type |");
362                 Console.WriteLine("-----");
363                 break;
364
365             case fieldMenu.ADDRESS:
366                 //Search the address field.
367                 Console.WriteLine("-----");
368                 Console.WriteLine("| Search Items -- Address |");
369                 Console.WriteLine("-----");
370                 break;
371
372             case fieldMenu.CITY:
373                 //Search the city field.
374                 Console.WriteLine("-----");
375                 Console.WriteLine("| Search Items -- City |");
376                 Console.WriteLine("-----");
377                 break;
378
379             case fieldMenu.PHONE:
380                 //Search the phone number field.
381                 Console.WriteLine("-----");
382                 Console.WriteLine("| Search Items -- Phone Number |");
383                 Console.WriteLine("-----");
384                 break;

```

```

385
386         case fieldMenu.BACK:
387             //Nothing to do; the user wants to go back.
388         default:
389             //Catch-all.
390             return choice;
391     }
392
393     //Ask for the search text.
394     Console.Write("Enter your search text: ");
395
396     //Get the user's search text and pipe that directly into the search method.
397     CViewDataSet foundData = data.Search(Console.ReadLine(), (CViewData.Fields)choice);
398
399     //Show the number of items found.
400     Console.WriteLine("{0} item{1} found.\n", foundData.Count,
401         foundData.Count == 1 ? "" : "s");
402
403     //If any items found, display them.
404     if (foundData.Count != 0)
405         Console.WriteLine("{0}\n{1}", foundData.Header, foundData);
406
407     //Return choice so the calling method knows what the choice was and can act accordingly.
408     return choice;
409 }
410
411 /*-----
412  * Method:   FieldMenuDisplay
413  * Purpose:  Display the field menu and get a choice. Must have valid input to return.
414  * Input:    Nothing.
415  * Output:   fieldMenu, representing the choice that was made.
416  -----*/
417 private fieldMenu FieldMenuDisplay()
418 {
419     fieldMenu menuChoice = 0;
420     bool invalid = true;
421
422     do
423     {
424         //Display the menu.
425         Console.WriteLine("Please select the field you would like to work with:");
426         Console.WriteLine(" 1) Facility Name");
427         Console.WriteLine(" 2) Facility Type");
428         Console.WriteLine(" 3) Street Address");
429         Console.WriteLine(" 4) City");
430         Console.WriteLine(" 5) Phone Number");
431         Console.WriteLine(" 6) Back");
432         Console.Write("Choice: ");
433
434         //Get the user's choice.
435         string input = Console.ReadLine();
436
437         //Extra line for formatting.
438         Console.WriteLine();
439
440         //Validate the user input.
441         invalid = !fieldMenu.TryParse(input, out menuChoice) ||
442             !FieldMenuValidate(menuChoice);
443     } while (invalid);
444
445     //Return the user's choice.
446     return menuChoice;
447 }
448

```

```
449      /*-----
450      * Method: FieldMenuValidate
451      * Purpose: Validates that the choice by the user is within the limits and is logically
452      *           possible.
453      * Input:   mmodifyMenu value, contains the user's choice.
454      * Output:  bool, representing whether the user's choice was valid or not.
455      -----*/
456      private bool FieldMenuValidate(fieldMenu value)
457      {
458          //Check to make sure that the user input is within valid limits.
459          if (value < FIELDMENU_MIN || value > FIELDMENU_MAX)
460              return false;
461
462          //Otherwise, input is good.
463          return true;
464      }
465
466      /*-----
467      * Method: MainMenuAction
468      * Purpose: Acts on the user's choice made at the Main Menu.
469      * Input:   mainMenu choice, represents the action specified.
470      * Output:  mainMenu, represents the action specified.
471      -----*/
472      private mainMenu MainMenuAction(mainMenu choice)
473      {
474          //Decide what to do based on the user's choice.
475          switch (choice)
476          {
477              case mainMenu.ADD:
478                  //Add a new item.
479                  DataAdd();
480                  break;
481
482              case mainMenu.MODIFY:
483                  //Modify an existing item.
484                  DataModify();
485                  break;
486
487              case mainMenu.SEARCH:
488                  //Search items.
489                  DataSearch();
490                  break;
491
492              case mainMenu.DELETE:
493                  //Delete an item.
494                  DataDelete();
495                  break;
496
497              case mainMenu.DISPLAY_ALL:
498                  //Display all the items.
499                  DataDisplayAll();
500                  break;
501
502              case mainMenu.EXIT:
503                  //Do nothing, exiting the method.
504              default:
505                  //Catch-all.
506                  break;
507          }
508
509          //Return choice so the calling method knows what the choice was and can act accordingly.
510          return choice;
511      }
512
```



```

513     /*-----
514     * Method: MainMenuDisplay
515     * Purpose: Display the main menu and get a choice. Must have valid input to return.
516     * Input:  Nothing.
517     * Output: mainMenu, representing the choice that was made.
518     -----*/
519     private mainMenu MainMenuDisplay()
520     {
521         mainMenu menuChoice = 0;
522         bool invalid = true;
523
524         do
525         {
526             //Display the menu.
527             Console.WriteLine("-----");
528             Console.WriteLine("| Main Interactive Menu |");
529             Console.WriteLine("-----");
530             Console.WriteLine("Please select an option:");
531             Console.WriteLine(" 1) Add New Item");
532             Console.WriteLine(" 2) Modify Item");
533             Console.WriteLine(" 3) Search Items");
534             Console.WriteLine(" 4) Delete Item");
535             Console.WriteLine(" 5) Display All Items");
536             Console.WriteLine(" 6) Exit");
537             Console.Write("Choice: ");
538
539             //Get the user's choice.
540             string input = Console.ReadLine();
541
542             //Extra line for formatting.
543             Console.WriteLine();
544
545             //Validate the user input.
546             invalid = !mainMenu.TryParse(input, out menuChoice) ||
547                     !MainMenuValidate(menuChoice);
548         } while (invalid);
549
550         //Return the user's choice.
551         return menuChoice;
552     }
553
554     /*-----
555     * Method: MainMenuValidate
556     * Purpose: Validates that the choice by the user is within the limits and is logically
557     *           possible.
558     * Input:   mainMenu value, contains the user's choice.
559     * Output:  bool, representing whether the user's choice was valid or not.
560     -----*/
561     private bool MainMenuValidate(mainMenu value)
562     {
563         //Check to make sure that the user input is within valid limits.
564         if (value < MAINMENU_MIN || value > MAINMENU_MAX)
565             return false;
566
567         //If the data set is empty, limit user to adding an entry or exiting.
568         if (data.Count == 0 && (value != mainMenu.ADD && value != mainMenu.EXIT))
569         {
570             Console.WriteLine("No data is present. Please choose a different option.\n");
571             return false;
572         }
573
574         //Otherwise, input is good.
575         return true;
576     }

```

```
577     }  
578 }  
579
```