```cpp
/*************************************************************
* AUTHOR     : Amirarsalan Valipour
* STUDENT ID     : 1103126
* Assignment #5  : DVD Movie ListIntro to OOP
* CLASS          : CS 1B
* SECTION        : MW - 7:30 pm - 9:50 pm
* DUE DATE       : 12/16/2019
*************************************************************/

#include "MovieList.h"

                /*********************************
                 *   CONSTRUCTOR / DESTRUCTOR   *
                 *********************************/

/*************************************************************************
 * MovieList ();
 * Constructor; Initialize class attributes
 * Parameters: none
 * Return: none
 ************************************************************************/

MovieList :: MovieList()
{

}


/*************************************************************************
 * ~MovieList ();
 * Destructor; does not perform any specific function
 * Parameters: none
 * Return: none
 ************************************************************************/

MovieList :: ~MovieList()
{

}


                        /*******************
                         *    MUTATORS    *
                         *******************/

```

```
46 /*****************************************************************************
*
47  * void CreateList (string inputFileName);
48  *
49  * Mutator; This method will create a movie list using the input
file
50  *           data
51  *
-------------------------------------------------------------------
52  * Parameter: inFileName (string)
53  *
-------------------------------------------------------------------
54  * Return: none
55
*****************************************************************************
*/
56
57 void MovieList :: CreateList(string iFileName)
58 {
59     ifstream inFile;//CALC & OUT - OSTREAM FOR INPUT FILE
60
61     DVDNode inputPtr;   //CALC - POINTER FOR EACH INPUT VARIABLE
62
63
64     //CHECKS FOR DEFAULT FILE NAME
65
66     if(iFileName == "d")
67     {
68         inFile.open("InputFile.txt");
69     }
70
71     else
72     {
73         inFile.open(iFileName);
74     }
75
76
77     //INPUT LINE BY LINE
78
79     while(inFile)
80     {
81         getline(inFile, inputPtr.title);
82
83         getline(inFile, inputPtr.leadingActor);
```

```
 84
 85         getline(inFile, inputPtr.supportingActor);
 86
 87         getline(inFile, inputPtr.genre);
 88
 89         getline(inFile, inputPtr.alternateGenre);
 90
 91         inFile >> inputPtr.year;
 92
 93         inFile >> inputPtr.rating;
 94
 95         inFile.ignore(1000, '\n');
 96
 97         getline(inFile, inputPtr.synopsis);
 98
 99         inFile.ignore(1000, '\n');
100
101         StackList :: Push(inputPtr);
102     }
103
104     //CLOSE FILE
105
106     inFile.close();
107 }
108
109
110                         /*******************
111                          *    ACCESSORS    *
112                          *******************/
113
114 /*************************************************************************
115  * void OutputList (string oFileName) const;
116  *
117  * Accessor; This method will output the list into the output file
118  * ----------------------------------------------------------------------
119  * Parameters: oFileName (string)
120  * ----------------------------------------------------------------------
121  * Return: none
122  ************************************************************************/
123
124 void MovieList :: OutputList(string oFileName) const
125 {
126     ofstream oFile;          //CALC & OUT - OSTREAM FOR OUTPUT FILE
127     int  i;              //CALC       - LCV
128     string   outPlot;        //CALC & OUT - ADJUSTED PLOT
```

```
129     DVDNode  *mvPtr;    //CALC       _ POMITER FOR OUTPUT
130
131
132     //CHECKS FOR DEFAULT FILE NAME
133
134     if (oFileName == "d")
135     {
136         oFile.open("OutputFile.txt");
137     }
138
139     else
140     {
141         oFile.open(oFileName);
142     }
143
144
145     //INITIALIZING
146
147     i = 0;
148
149     mvPtr = NULL;
150
151     mvPtr = head;
152
153
154     //OUTPUT HEADER
155
156     PrintHeader(oFile, "DVD Movie ListIntro to OOP", 5, 'A');
157
158
159     //PROCESSING
160
161     while(mvPtr != NULL)
162     {
163         i++;
164
165         //ADJUST AND ALTER EACH OUTPUT
166
167         mvPtr -> title = StringAdj(mvPtr -> title, 47);
168
169         mvPtr -> genre = StringAdj(mvPtr -> genre, 25);
170
171         mvPtr -> alternateGenre =
172               StringAdj(mvPtr -> alternateGenre, 25);
173
```

```
174         mvPtr -> leadingActor =
175               StringAdj(mvPtr -> leadingActor, 17);
176
177         mvPtr -> supportingActor=
178               StringAdj(mvPtr -> supportingActor,17);
179
180
181         //OUTPUT INTO THE FILE
182
183         oFile << left;
184
185         oFile << "**************************************"
186               "**************************************\n";
187
188         oFile << "MOVIE #: " << setw(9) << i
189           << "Title: " << mvPtr -> title << endl;
190
191         oFile << "--------------------------------------"
192               "--------------------------------------\n";
193
194         oFile << "Year: " << setw(12) << mvPtr -> year
195           << "Rating: " << mvPtr -> rating << endl;
196
197         oFile << "--------------------------------------"
198               "--------------------------------------\n";
199
200         oFile << setw(18) << "Leading actor: " << setw(26)
201           << mvPtr -> leadingActor << "Genre 1: " << mvPtr -> genre
    << endl;
202
203         oFile << setw(18) << "Supporting actor: " << setw(26)
204           << mvPtr -> supportingActor << "Genre 2: "
205               << mvPtr -> alternateGenre << endl;
206
207         oFile << "--------------------------------------"
208               "--------------------------------------\n";
209
210         oFile << "PLOT:" << endl;
211
212         outPlot = WordAdj(mvPtr -> synopsis);
213
214         oFile << outPlot;
215
216         oFile << "**************************************"
217               "**************************************\n";
```

```
218
219         oFile << endl << endl;
220
221         oFile << right;
222
223         mvPtr = mvPtr -> next;
224
225     }
226
227     //CLOSE OUTPUT FILE
228
229     oFile.close();
230
231 }
232
233
234 /*****************************************************************
235 * string WordAdj (string plot) const;
236 *
237 * Accessor; This method adjusts the string and the size of the
    words
238 * ---------------------------------------------------------------
239 * Parameters: plot (string)
240 * ---------------------------------------------------------------
241 * Return: returnStr (string)
242 *****************************************************************/
243
244 string MovieList :: WordAdj (string plot) const
245 {
246     int i;                          //CALC - LCV
247
248     int size;                       //CALC - LCV
249
250     const int maxLength = 75;    //CALC - LCV
251
252     string alteredStr;              //CALC & OUT - ALTERED STRING
253     string tempLine;          //CALC - TEMPORARY LINE
254     string tempWord;          //CALC - TEMPORARY WORD
255
256
257     //INITIALIZING
258
259     size = plot.length();
260
261     alteredStr = "";
```

```
262
263     tempLine = "";
264
265     tempWord = "";
266
267
268     //PROCESSING
269
270     for (i = 0; i < size; i++)
271     {
272         if (plot[i] != ' ')
273         {
274             tempWord = tempWord + plot[i];
275         }
276
277         else
278         {
279             if (tempLine.length() + tempWord.length() > maxLength)
280             {
281                 alteredStr = alteredStr + tempLine + '\n';
282
283                 tempLine.clear();
284             }
285
286             tempLine = tempLine + tempWord + " ";
287
288             tempWord.clear();
289         }
290
291     }
292
293     if(tempLine != "")
294     {
295         alteredStr = alteredStr + tempLine + tempWord + '\n';
296     }
297
298     return alteredStr;
299 }
300
301
302 /*****************************************************************
303  * FUNCTION PrintHeader
304  *_____________________________________________________________
305  * This function receives an assignment name, type
306  *   and number then outFiles the appropriate header -
```

```
 *   returns nothing.
 *_____________________________________________________________
 * PRE-CONDITIONS
 *
 *      outFile: Ostream variable
 *      asName : Assignment Name has to be previously defined
 *      asType : Assignment Type has to be previously defined
 *      asNum  : Assignment Number has to be previously defined
 *
 * POST-CONDITIONS
 *
 *      This function will output the class heading.
 ***************************************************************/

void MovieList :: PrintHeader(ostream &outFile,
                              string  asName,
                              int     asNum,
                              char    asType) const
{
    const int  PROMPT = 14;

    const char PROGRAMMER[25] = "Amirarsalan Valipour";
    const char CLASS[5]       = "CS1B";
    const char SECTION[25]    = "MW: 7:30p - 9:50p";

    outFile << left;
    outFile << endl;
    outFile <<
"*****************************************************************";
    outFile << "\n* PROGRAMMED BY : " << PROGRAMMER;
    outFile << "\n* " << setw(PROMPT) << "CLASS"   << ": " <<
CLASS;
    outFile << "\n* " << setw(PROMPT) << "SECTION" << ": " <<
SECTION;
    outFile << "\n* ";

    if (toupper(asType) == 'L')
    {
        outFile << "LAB #" << setw(8);
    }

    else
    {
        outFile << "ASSIGNMENT #" << setw(1);
    }
```

```
349
350     outFile << asNum << " : " << asName;
351     outFile <<
    "\n**************************************************";
352     outFile << "**\n\n";
353     outFile << right;
354
355 }
356
357
358 /
    ************************************************************************
    ***********
359  * FUNCTION StringAdj
360  *
    ------------------------------------------------------------------------
    ---------
361  * This function will setup the length for the movie plot.
362  *
    ------------------------------------------------------------------------
    ---------
363  * PRE-CONDITIONS
364  *
365  *       string prompt : original string
366  *       int maxLength : maximum length
367  *
368  * POST-CONDITIONS
369  *
370  *      Returns new adjusted string to fit the length
371
    ************************************************************************
    ***********/
372
373 string MovieList :: StringAdj(string plot,int maxLength) const
374 {
375     int i;              //CALC - LCV
376
377     string plotStr;     //CALC & OUT - ADJUSTED STRING
378
379     bool maxed;         //CALC - LCV
380
381     i = 0;
382
383     //INITIALIZE
384
```

```
385     plotStr = "";
386
387     maxed = false;
388
389     //PROCESSING
390
391     while (((unsigned) i < plot.length()) && !maxed)
392     {
393         if (plotStr.length() <= (unsigned) maxLength)
394         {
395             plotStr = plotStr + plot[i];
396         }
397
398         else
399         {
400                 maxed = true;
401                 plotStr = plotStr.substr(0, maxLength - 3) + "...";
402         }
403
404         i++;
405     }
406
407     //RETURN
408
409     return plotStr;
410 }
411
412
```