

## StackList.cpp

```
1 /*****
2 * AUTHOR      : Amirarsalan Valipour
3 * STUDENT ID   : 1103126
4 * Assignment #5 : DVD Movie ListIntro to OOP
5 * CLASS       : CS 1B
6 * SECTION     : MW - 7:30 pm - 9:50 pm
7 * DUE DATE    : 12/16/2019
8 *****/
9
10 #include "StackList.h"
11
12             /*****
13             *   CONSTRUCTOR / DESTRUCTOR   *
14             *****/
15
16
17 /*****
18 * StackList ();
19 * Constructor: Initialize class attributes
20 * Parameters: none
21 * Return:     none
22 *****/
23
24 StackList :: StackList()
25 {
26     head = NULL;
27
28     stackCount = 0;
29 }
30
31
32 /*****
33 * ~StackList ();
34 * Destructor: does not perform anything
35 * Parameters: none
36 * Return: none
37 *****/
38
39 StackList :: ~StackList()
40 {
41     DVDNode *stckPtr;
42
43     stckPtr = head;
44
45     while (stckPtr != NULL)
```

## StackList.cpp

```
46     {
47         head = head -> next;
48
49         delete stckPtr;
50
51         stckPtr = head;
52     }
53 }
54
55
56         /*****
57         *      MUTATORS      *
58         *****/
59
60 /
61 ****
62 * void Push (DVDNode newDVD);
63 *
64 * Mutator: This method will add a DVD node to the list to the
65 front
66 *
67 -----
68 * Parameter: newDVD (DVDNode) //IN - node to be added to list
69 *
70 -----
71 * Return: none
72
73 ****
74 */
75
76 void StackList :: Push (DVDNode newDVD)
77 {
78     DVDNode *persPtr;
79
80     persPtr = head;
81
82     persPtr = new DVDNode;
83
84     *persPtr = newDVD;
85
86     //QUEUE
87
88     persPtr -> next = head; head = persPtr;
89 }
```

## StackList.cpp

```
85
86 //UPDATE COUNTER
87
88 stackCount++;
89
90 persPtr = NULL;
91 }
92
93
94 /
*****
95 * DVDNode Pop ();
96 *
97 * Mutator: This method will remove a DVD node from the front of
the
98 * list and return the DVDNode being removed.
99 *
-----
100 * Parameter: none
101 *
-----
102 * Return: dvdPtr (DVDNode)
103
*****
*/
104
105 DVDNode StackList :: Pop ()
106 {
107     DVDNode dvdPtr;
108
109     DVDNode *persPtr;
110
111     persPtr = head;
112
113     if(IsEmpty())
114     {
115         cout << "The stack is empty!";
116
117         return dvdPtr;
118     }
119
120     dvdPtr = Peek();
121
122     persPtr = persPtr -> next;
123
```

## StackList.cpp

```
124     delete persPtr;
125
126     persPtr = NULL;
127
128     return dvdPtr;
129
130 }
131
132
133                                     /*****
134                                     *   ACCESSORS   *
135                                     *****/
136
137 /*****
138 * bool IsEmpty () const;
139 *
140 * Accessor; This method will return the boolean value whether
141 *           the list is empty or not empty.
142 * -----
143 * Parameters: none
144 * -----
145 * Return: emptyCheck (bool)
146 *****/
147
148 bool StackList :: IsEmpty() const
149 {
150     bool emptyCheck;
151
152     if (stackCount == 0)
153     {
154         emptyCheck = true;
155     }
156
157     else
158     {
159         emptyCheck = false;
160     }
161
162     return emptyCheck;
163 }
164
165
166 /*****
167 * DVDNode Peek () const;
168 *
```

## StackList.cpp

```
169 * Accessor; This method will return the DVD node of the first
170 *           element on the list
171 * -----
172 * Parameters: none
173 * -----
174 * Return: dvdPtr (DVDNode)
175 *****/
176
177 DVDNode StackList :: Peek() const
178 {
179     DVDNode dvdPtr;
180
181     dvdPtr.title = "EMPTY";
182
183     if(!IsEmpty())
184     {
185         dvdPtr = *head;
186     }
187
188     return dvdPtr;
189 }
190
191
192 /*****
193 * int Size () const;
194 *
195 * Accessor; This method will return the size of the list
196 * -----
197 * Parameters: none
198 * -----
199 * Return: stackCount (int)
200 *****/
201 int StackList :: Size() const
202 {
203     return stackCount;
204 }
205
206
207
208
```